

NPS-ME-99-007

NAVAL POSTGRADUATE SCHOOL

Monterey, California



IMPLEMENTATION OF A SHELL ELEMENT WITH PRESSURE AND VOID EFFECTS INTO DYSMAS

by

Patrick M. McDermott

and

Young W. Kwon

September 1999

Approved for public release; distribution is unlimited.

DTIC QUALITY INSPECTED 4

19991122 086

Naval Postgraduate School
Monterey, California

RADM Robert C. Chaplin
Superintendent

R.S. Elster
Provost

This report was prepared in conjunction with research conducted for Naval Surface Warfare Center, Carderock Division.

This report was prepared by:



Patrick M. McDermott
Lieutenant, United States Navy

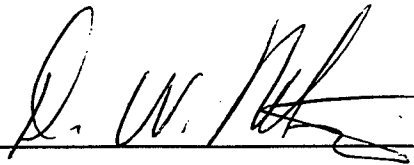


Young W. Kwon
Associate Professor of Mechanical Engineering



Reviewed By:

for M. D. Kelleher, Acting Chair
Department of Mechanical Engineering



Released By:

D. W. Netzer
Associate Provost and
Dean of Research

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY (Leave blank)

2. REPORT DATE
September 1999

3. REPORT TYPE AND DATES COVERED

4. TITLE AND SUBTITLE

**IMPLEMENTATION OF A SHELL ELEMENT WITH PRESSURE
AND VOID EFFECTS INTO DYSMAS**

5. FUNDING NUMBERS

N0016799WX90504

6. AUTHOR(S)

McDermott, Patrick M. and Kwon, Young W.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION
REPORT NUMBER

NPS-ME-99-007

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Naval Surface Warfare Center, Carderock Division
West Bethesda, Maryland 20817-5700

10. SPONSORING / MONITORING
AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

A

13. ABSTRACT (maximum 200 words)

A shell formulation was developed from a three-dimensional solid. The shell element has four corner nodes at which there are three displacements and three rotations as nodal degrees of freedom, and includes both transverse shear and transverse normal deformations. The element utilizes reduced integration along the in-plane axes and full integration along the transverse axis. The formulation incorporates the Gurson constitutive model for void growth and plastic deformation. An algorithm for stable solutions of the nonlinear constitutive equations is also developed. Hourglass mode control is provided by adding a small fraction of internal force determined through full integration along the in-plane axes and reduced integration along the transverse axis. Implementation into both a specialized research finite element program and DYSMAS, a derivative of DYNA3D, is discussed. Numerical examples are provided to verify the accuracy of the new element and to show the importance of the transverse normal stress, void effects on plastic strain, and the necessity of applying a drilling moment.

14. SUBJECT TERMS

Finite Element, Shell Formulation, Void Model, Elasto-plastic, DYSMAS,
DYNA3D, High Order Element

15. NUMBER OF
PAGES

112

16. PRICE CODE

17. SECURITY
CLASSIFICATION OF
REPORT

Unclassified

18. SECURITY CLASSIFICATION OF
THIS PAGE

Unclassified

19. SECURITY CLASSIFI- CATION
OF ABSTRACT

Unclassified

20. LIMITATION OF
ABSTRACT

UL

ABSTRACT

A shell formulation was developed from a three-dimensional solid. The shell element has four corner nodes at which there are three displacements and three rotations as nodal degrees of freedom, and includes both transverse shear and transverse normal deformations. The element utilizes reduced integration along the in-plane axes and full integration along the transverse axis. The formulation incorporates the Gurson constitutive model for void growth and plastic deformation. An algorithm for stable solutions of the nonlinear constitutive equations is also developed. Hourglass mode control is provided by adding a small fraction of internal force determined through full integration along the in-plane axes and reduced integration along the transverse axis. Implementation into both a specialized research finite element program and DYSMAS, a derivative of DYNA3D, is discussed. Numerical examples are provided to verify the accuracy of the new element and to show the importance of the transverse normal stress, void effects on plastic strain, and the necessity of applying a drilling moment.

TABLE OF CONTENTS

I. INTRODUCTION.....	1
II. FINITE ELEMENT FORMULATION.....	5
A. GEOMETRY	5
B. DISPLACEMENT.....	7
C. COORDINATE TRANSFORMATION.....	8
D. STRAIN DISPLACEMENT RELATION.....	10
E. JACOBIAN MATRIX.....	11
F. STRESS-STRAIN RELATIONSHIP.....	12
G. DRILLING MOMENTS.....	12
H. INTERNAL FORCE, MASS AND THEIR ASSEMBLY.....	14
I. EXPLICIT TIME INTEGRATION	15
III. DAMAGE CONSTITUTIVE EQUATIONS.....	17
A. GURSON'S VOID MODEL.....	17
B. IMPROVING STABILITY IN THE CONSTITUTIVE EQUATIONS.....	20
C. STRAIN HARDENING.....	22
IV. HOURGLASS MODE CONTROL	25
VI. NUMERICAL EXAMPLES.....	31
A. ELASTIC PLATE.....	31
B. THICK CLAMPED PLATE UNDER PRESSURE LOAD IN ELASTO-PLASTIC REGION	32
C. THICK CLAMPED PLATE WITH CENTRAL POINT LOAD IN THE ELASTO-PLASTIC REGION.....	37
D. SIMPLY SUPPORTED PLATE WITH CENTRAL POINT LOAD IN THE ELASTO-PLASTIC REGION.....	40
E. ELASTIC PINCHED CYLINDER.....	42
F. THICK PINCHED CYLINDER IN THE ELASTO-PLASTIC REGION.....	42
G. ELASTIC SPHERICAL CAP WITH A CENTER HOLE	46
H. SPHERICAL CAP WITH A CENTER HOLE, IMPACT LOADING.....	48
I. SPHERICAL CAP WITH A CENTER HOLE, ELASTO-PLASTIC LOADING.....	49
V. DYSMAS IMPLEMENTATION	57
A. GENERAL IMPLEMENTATION ISSUES.....	57
B. SHELL FORMULATION.....	58
C. MATERIAL MODEL.....	59
D. IMPLEMENTATION ISSUES.....	60
VI. DYSMAS VERIFICATION	63
A. ELASTIC CANTILEVER PLATE WITH SMALL DISPLACEMENT	63
B. SINGLE CURVATURE VERIFICATION: ELASTIC-PLASTIC CYLINDER.....	65
C. BALL IMPACT PROBLEM	68
VII. CONCLUSIONS AND RECOMMENDATIONS.....	73
APPENDIX A. CORRECTIONS IMPLEMENTED INTO DYSMAS.....	75
APPENDIX B. NEW DYSMAS SUBROUTINE LISTINGS.....	77
APPENDIX C. DYSMAS MODIFIED SUBROUTINE LISTINGS.....	119

APPENDIX D. DOCUMENTATION PAGE FOR NEW MATERIAL TYPE	159
LIST OF REFERENCES	161
INITIAL DISTRIBUTION LIST	165

LIST OF FIGURES

Figure 1. Element Cross Section.....	5
Figure 2. Displacement Vector Orientation.....	7
Figure 3. Calculating a New Yield Stress.....	23
Figure 4. Hourglass Modes in a Pinched Cylinder Model.....	25
Figure 5. Hourglass Mode Control Integration Points.....	26
Figure 6. Pinched Cylinder Model with Hourglass Mode Control.....	28
Figure 7. Nine Element Clamped Plate Mesh.....	33
Figure 8. Void Effects in a Clamped Plate with Pressure Loading: Top (a) - No Void Effects, Bottom (b) - Void Effects.....	35
Figure 9. Transverse Normal Stress Variation through Shell Thickness: Clamped Plate with Pressure Loading and Void Effects.....	36
Figure 10. Stress Component Time History in Bottom Fiber: Clamped Plate with Concentrated Load, Void Effects, and Drilling Moment Applied.....	38
Figure 11. Porosity versus Effective Plastic Strain in Bottom Fiber: Clamped Plate with Concentrated Load, Void Effects, and Drilling Moment Applied.....	39
Figure 12. Mesh Structure for Pinched Cylinder.....	44
Figure 13. Stress Component Time History in Bottom Fiber: Pinched Cylinder, No Void Effects or Drilling Moment.....	45
Figure 14. Stress Component Time History in Bottom Fiber: Pinched Cylinder, Void Effects and Drilling Moment Applied.....	46
Figure 15. Mesh Structure for Spherical Cap.....	48
Figure 16. Node One Displacement Time History: Pinched Spherical Cap with Impact Loading.....	49
Figure 17. Porosity versus Effective Plastic Strain in Inner Fiber: Spherical Cap with Void and Drilling Moment Effects.....	51
Figure 18. Comparison of Void and Drilling Moment Effects in a) Compression (Top) and b) Tension (Bottom) for Spherical Cap.....	52
Figure 19. Contact Node Mean Displacement Time History: Spherical Cap with Void and Drilling Moment Effects.....	53
Figure 20. Contact Node Displacement: Spherical Cap with 5% Greater Load.....	54
Figure 21. Deformed Structure at End of Analysis: Spherical Cap with 5% Greater Load.	55
Figure 22. Comparison of Element Formulations for Cantelever Plate.....	65
Figure 23. Mesh Structure for Pinched Cylinder using TECPLOT.....	66
Figure 24. Elastic-Plastic Pinched Cylinder using DYSMAS.....	67
Figure 25. Ball Impact Problem Center Node Displacement with 5:1 Aspect Ratio.....	68
Figure 26. Ball Impact Problem with Plate Modeled using Solid Elements.....	69
Figure 27. Ball Impact Problem with Shell Elements.....	70
Figure 28. Center Node Displacement Comparison for Ball Impact Problem.....	71

LIST OF TABLES

Table 1. Comparison of Results for Elastic Clamped Plate.....	32
Table 2. Material Properties of Clamped Plate.....	32
Table 3. Void Characteristics of Clamped Plate.....	32
Table 4. Summary of Results for Clamped Plate with Pressure Load.....	37
Table 5. Summary of Results for Clamped Plate Subjected to Point Force.....	40
Table 6. Summary of Results for a Simply Supported Plate with a Point Force.....	41
Table 7. Comparison of Results for Elastic Pinched Cylinder.....	42
Table 8. Summary of Results for Elasto-Plastic Pinched Cylinder.....	43
Table 9. Comparison of Results for Spherical Cap with Elastic Loading, and Results from MacNeal and Harder [26].....	47
Table 10. Summary of Results for Elasto-Plastic Spherical Cap.....	51
Table 11. DYSMAS Solution of Cantilever Plate.....	64
Table 12. DYSMAS Elastic Pinched Cylinder Results.....	67

I. INTRODUCTION

Since shell structures are efficient load-carrying structural members, they have been dominant in most structural applications. However, the finite element formulation of shells poses some difficulty. As a result, extensive study has been devoted to developing better shell elements. Because of the abundance of papers on this subject, no attempt is made here to summarize all of them. Some of the related past work is given in references [1-12].

Void growth and nucleation can have a significant effect on plastic flow [13]. Since voids act as stress concentrators, the overall effect is to reduce the stress under plastic flow, and increase the plastic strain [14-16]. The model proposed by Gurson appears to have been adopted as the standard for incorporating void growth and nucleation effects into a numerical solution of elasto-plastic problems [17]. Previous work has been done on improving the efficiency and accuracy of plasticity computations, applying plasticity to plate/shell elements, and incorporating void growth and nucleation effects in solid elements [18-20].

The element presented in this paper incorporates a modified version of the algorithm for three-dimensional solids proposed by Aravas [21] into a shell element. This shell element assumes a modified plane-stress condition, and utilizes both the hydrostatic pressure and the deviatoric stress. Due to the importance of the hydrostatic pressure on void growth and nucleation, this element includes the transverse normal stress.

The algorithm proposed by Aravas is not unconditionally stable when applied to this modified plane stress condition [21]. This paper also presents modifications to the original algorithm, which greatly enhance the stability of the solution, at the cost of a slight decrease in computational efficiency. This algorithm also allows for more complicated work-

hardening profiles than the typical exponent law ($\sigma = K\epsilon^n$). Full modeling of the entire stress-strain curve is accomplished by a piece-wise linear approximation. While this method is not particularly useful for analytic approaches, it appears to be helpful in strictly numerical solutions. The stress-strain relationship may be taken directly off the results of a standard tensile test.

For thick shell applications, the transverse normal stress and strain cannot be neglected. The transverse normal stress and strain affect both the hydrostatic pressure and the deviatoric stress, which in turn affect plastic deformation and void growth and nucleation. The element presented here extends the typical use of the drilling degree of freedom (DOF), as outlined in Hughes and Brezzi [22], to incorporate the effects of transverse normal strain. The drilling DOF is important in transmitting stress and strain to elements across sharp bends and curves, and is therefore already included in a variety of shell elements [2,4,6,7,10]. In addition to this traditional usage, the present element utilizes the drilling DOF to compute the transverse normal deformation. The importance of including the transverse normal deformation is outlined in Essenburg [23].

The present study formulates a shell element for transient analysis. The element can have elasto-plastic deformation with void growth and nucleation. Gurson's void model is used as a basis for the void constitutive model. The shell element includes both transverse shear deformation and the transverse normal deformation for thick shell applications. The drilling degree of freedom is used for computing the deformation through the thickness of a thick shell. An algorithm for stable solutions of the nonlinear constitutive equations is also

developed.

Some example problems are presented to evaluate the formulation and to investigate the effects of the transverse normal strain for thick shells in association with elasto-plastic deformation, including void effects. Implementation and verification for DYSMAS is then discussed.

II. FINITE ELEMENT FORMULATION

A. GEOMETRY

A point in a shell structure can be expressed by a vector sum of two vectors. The first vector is a position vector from the origin of the global coordinate system to a point on a reference surface of the shell element. The second vector is a position vector from this reference surface to the point under consideration. The surface that spans the center of the transverse axis is used as the reference surface in this formulation, although any surface would suffice. The first vector terminates at the reference surface directly below the point in question. The second vector is then the normal from the reference surface that intersects the desired point. Figure 1 shows this relationship. Two shape functions are used to describe

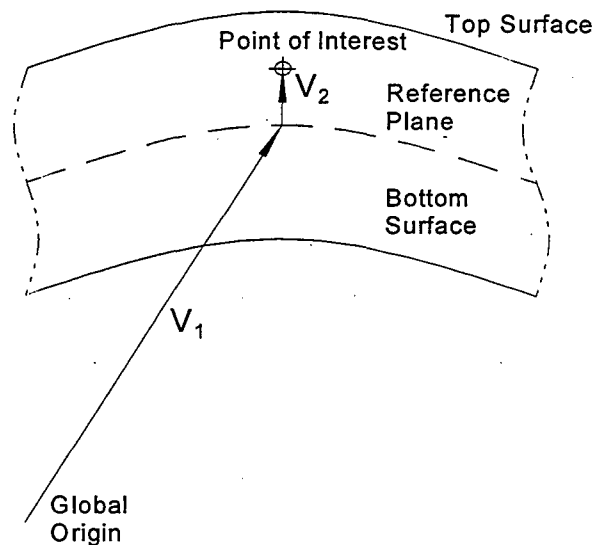


Figure 1. Element Cross Section.

a position in the element; N^k is the two dimensional shape function in the ξ - η plane, and H^k is the one dimensional shape function along the ζ axis, where (ξ, η, ζ) describes a point in the natural coordinate system. A generic point in the shell may now be described in terms of the position vectors of the nodes and the shape functions:

$$x_i(\xi, \eta, \zeta) = \sum_{k=1}^n N^k(\xi, \eta) x_i^k + \sum_{k=1}^n N^k(\xi, \eta) H^k(\zeta) V_{3i}^k \quad (i = 1, 2, 3) \quad (1)$$

where x_i^k is the position vector of node k in the reference surface; V_{3i}^k is the unit vector at the node k , and n is the number of nodes per element. In the present formulation, a four-node shell element is considered. The unit vector V_{3i}^k is defined as:

$$V_{3i}^k = \frac{(x_i^k)^{top} - (x_i^k)^{bottom}}{\|(x_i^k)^{top} - (x_i^k)^{bottom}\|} \quad (2)$$

where *top* and *bottom* indicate the top and bottom surfaces of the shell, and $\| \cdot \|$ denotes the Euclidean norm. The one-dimensional shape function H^k is expressed as:

$$H^k(\zeta) = \left[\frac{1}{4}(1 + \zeta)(1 - \bar{\zeta}) - \frac{1}{4}(1 - \zeta)(1 + \bar{\zeta}) \right] \|(x_i^k)^{top} - (x_i^k)^{bottom}\| \quad (3)$$

in which $\bar{\zeta}$ indicates the location of the reference surface and varied from -1 to 1 ($\bar{\zeta} = 0$ denotes the mid-surface). The two-dimensional shape function N^k is expressed as:

$$\begin{aligned} N^1 &= \frac{1}{4}(1 - \xi)(1 - \eta) \\ N^2 &= \frac{1}{4}(1 + \xi)(1 - \eta) \\ N^3 &= \frac{1}{4}(1 + \xi)(1 + \eta) \\ N^4 &= \frac{1}{4}(1 - \xi)(1 + \eta) \end{aligned} \quad (4)$$

B. DISPLACEMENT

The displacement field in a shell can be written as:

$$u_i(\xi, \eta, \zeta) = \sum_{k=1}^n N^k(\xi, \eta) u_i^k + \sum_{k=1}^n N^k(\xi, \eta) H^k(\zeta) (-V_{2i}^k \theta_1^k + V_{1i}^k \theta_2^k + V_{3i}^k \theta_3^k) \quad (i=1,2,3) \quad (5)$$

in which u_i is the displacement along the x_i axis, u_i^k is the nodal displacement at the node k , and unit vectors V_{1i}^k and V_{2i}^k lie along the reference surface. V_{1i}^k, V_{2i}^k and V_{3i}^k are mutually perpendicular. θ_1^k, θ_2^k and θ_3^k are rotational degrees of freedom along the unit vectors V_{1i}^k, V_{2i}^k and V_{3i}^k , respectively. The right-hand rule is assumed for the positive direction of each rotation. Figure 2 illustrates the relationship among these vectors.

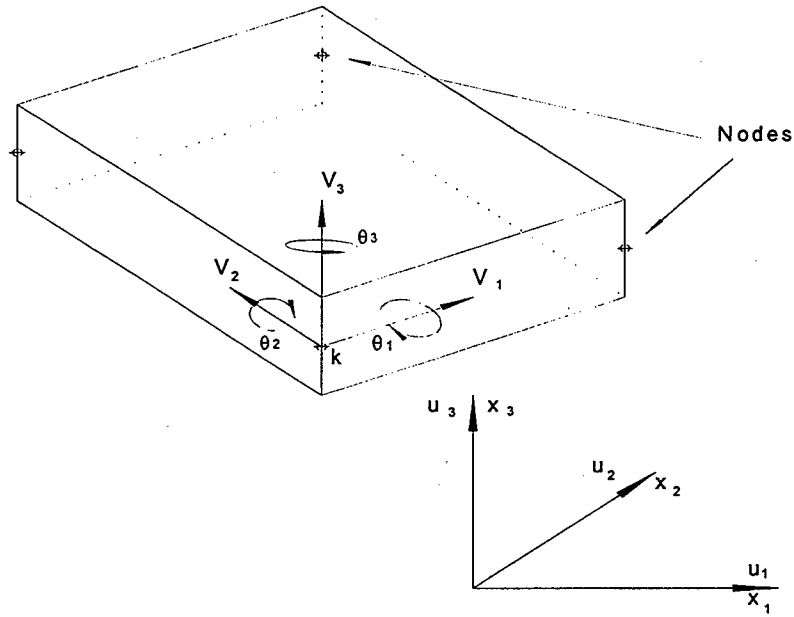


Figure 2. Displacement Vector Orientation.

θ_1^k and θ_2^k are the bending rotations, while θ_3^k is the drilling rotational degree of freedom. The role of θ_3^k can be clarified by considering a flat plate parallel to the x_1 - x_2 plane. Now Eq. (5) can be rewritten for the transverse displacement as:

$$u_3 = \sum_{k=1}^n N^k u_3^k + \sum_{k=1}^n N^k H^k \theta_3^k \quad (6)$$

Equation (6) demonstrates that the transverse deformation varies through the plate thickness (i.e. along the ζ axis) with θ_3^k . In this way, the transverse normal deformation is included in this formulation, along with the transverse shear deformations.

C. COORDINATE TRANSFORMATION

Combining the three unit direction vectors into matrix $[\mathbf{T}_p]$ provides the rotation transformation matrix, or matrix of direction cosines, as shown in Eq. (7). $[\mathbf{T}_p]^{-1}$ is used to transform the nodal degrees of freedom from the global coordinate system to local coordinates, as shown in Eq. (8), where k is the node number. The components of $\{d\}$ at the four nodes of an element are shown in Eq. (9). Once the internal force vector is generated in local coordinates, $[\mathbf{T}_p]$ is used to transform the local vectors back into global coordinates. This procedure will be discussed later.

$$[\mathbf{T}_p] = [\bar{\mathbf{v}}_1 \ \bar{\mathbf{v}}_2 \ \bar{\mathbf{v}}_3]_{(3 \times 3)} \quad (7)$$

$$\{d^k\}_{(6 \times 1)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & & & \\ 0 & 0 & 0 & [T_p]^{-1} & & \\ 0 & 0 & 0 & & & \end{bmatrix} \{d_{\text{global}}^k\}_{(6 \times 1)} \quad (k=1,2,3,4) \quad (8)$$

$$\{d\} = \{d^1 d^2 d^3 d^4\}^T \quad (9)$$

The strain transformation matrix, $[T]$, is used to transform calculated strain from the global coordinate system to local coordinates. Transforming the resulting stress from local coordinates to the global coordinate system would normally require using $[T]^{-1}$, but since $[T]$ is orthogonal, $[T]^{-1} = [T]^T$, where $[T]^T$ is the transpose of $[T]$. This property negates the requirement to invert a six by six matrix. For a detailed derivation of these transformations, refer to Cook [24]. The strain transformation matrix is explicitly defined in Eq. (10), where V_{ij} is the cosine of the direction vector V_i in the x_j direction.

$$[T] = \begin{bmatrix} V_{11}^2 & V_{12}^2 & V_{13}^2 & V_{11} V_{12} & V_{12} V_{13} & V_{11} V_{13} \\ V_{21}^2 & V_{22}^2 & V_{23}^2 & V_{21} V_{22} & V_{22} V_{23} & V_{21} V_{23} \\ V_{31}^2 & V_{32}^2 & V_{33}^2 & V_{31} V_{32} & V_{32} V_{33} & V_{31} V_{33} \\ 2 V_{11} V_{21} & 2 V_{12} V_{22} & 2 V_{13} V_{23} & V_{11} V_{22} + V_{21} V_{12} & V_{12} V_{23} + V_{22} V_{13} & V_{13} V_{21} + V_{23} V_{11} \\ 2 V_{21} V_{31} & 2 V_{22} V_{32} & 2 V_{23} V_{33} & V_{21} V_{32} + V_{31} V_{22} & V_{22} V_{33} + V_{32} V_{23} & V_{23} V_{31} + V_{33} V_{21} \\ 2 V_{11} V_{31} & 2 V_{12} V_{32} & 2 V_{13} V_{33} & V_{11} V_{32} + V_{31} V_{12} & V_{12} V_{33} + V_{32} V_{13} & V_{13} V_{31} + V_{33} V_{11} \end{bmatrix} \quad (10)$$

D. STRAIN-DISPLACEMENT RELATION

The six components of the strain tensor are computed from Eq. (5) by taking its derivative with respect to the x_i axis. In matrix form, the result for a four-node element is:

$$\{\varepsilon\} = [\mathbf{B}]\{d\} \quad (11)$$

$$\{\varepsilon\} = \{\varepsilon_{11} \varepsilon_{22} \varepsilon_{33} \gamma_{12} \gamma_{23} \gamma_{13}\}^T \quad (12)$$

where

$$[\mathbf{B}] = [[\mathbf{B}^1][\mathbf{B}^2][\mathbf{B}^3][\mathbf{B}^4]] \quad (13)$$

The detailed expression for $[\mathbf{B}^k]$ is:

$$[\mathbf{B}^k] = \begin{bmatrix} \frac{\partial N^k}{\partial x_1} & 0 & 0 & -g_1^k V_{21}^k & g_1^k V_{11}^k & g_1^k V_{31}^k \\ 0 & \frac{\partial N^k}{\partial x_2} & 0 & -g_2^k V_{22}^k & g_2^k V_{12}^k & g_2^k V_{32}^k \\ 0 & 0 & \frac{\partial N^k}{\partial x_3} & -g_3^k V_{23}^k & g_3^k V_{13}^k & g_3^k V_{33}^k \\ \frac{\partial N^k}{\partial x_2} & \frac{\partial N^k}{\partial x_1} & 0 & -g_2^k V_{21}^k - g_1^k V_{22}^k & g_2^k V_{11}^k + g_1^k V_{12}^k & g_2^k V_{31}^k + g_1^k V_{32}^k \\ 0 & \frac{\partial N^k}{\partial x_3} & \frac{\partial N^k}{\partial x_2} & -g_k^k V_{22}^k - g_k^k V_{23}^k & g_k^k V_{12}^k + g_k^k V_{13}^k & g_k^k V_{32}^k + g_k^k V_{33}^k \\ \frac{\partial N^k}{\partial x_3} & 0 & \frac{\partial N^k}{\partial x_1} & -g_3^k V_{21}^k - g_1^k V_{23}^k & g_3^k V_{11}^k + g_1^k V_{13}^k & g_3^k V_{31}^k + g_1^k V_{33}^k \end{bmatrix} \quad (14)$$

in which

$$g_i^k = \frac{\partial N^k}{\partial x_i} H^k + N^k \frac{\partial H^k}{\partial x_i} \quad (15)$$

The vector $\{d^k\}$ is defined as:

$$\{d^k\} = \{u_1^k u_2^k u_3^k \theta_1^k \theta_2^k \theta_3^k\} \quad (16)$$

where u_i is the displacement along the x_i direction at node k , and θ_i is the rotational displacement about the x_i axis at node k . The matrix $[\mathbf{B}^k]$ must be calculated for each integration point.

E. JACOBIAN MATRIX

Computing the derivatives $\frac{\partial N^k}{\partial x_i}$ and $\frac{\partial H^k}{\partial x_i}$ requires the Jacobian matrix, defined as

$$[\mathbf{J}] = \begin{bmatrix} x_{1,\xi} & x_{2,\xi} & x_{3,\xi} \\ x_{1,\eta} & x_{2,\eta} & x_{3,\eta} \\ x_{1,\zeta} & x_{2,\zeta} & x_{3,\zeta} \end{bmatrix} \quad (17)$$

where

$$x_{i,\xi} = \frac{\partial x_i}{\partial \xi} = \sum_{k=1}^n \frac{\partial N^k}{\partial \xi} x_i + \sum_{k=1}^n \frac{\partial N^k}{\partial \xi} H^k V_{3i}^k \quad (i=1,2,3) \quad (18)$$

$$x_{i,\eta} = \frac{\partial x_i}{\partial \eta} = \sum_{k=1}^n \frac{\partial N^k}{\partial \eta} x_i + \sum_{k=1}^n \frac{\partial N^k}{\partial \eta} H^k V_{3i}^k \quad (i=1,2,3) \quad (19)$$

$$x_{i,\zeta} = \frac{\partial x_i}{\partial \zeta} = \sum_{k=1}^n N^k \frac{\partial H^k}{\partial \zeta} V_{3i}^k \quad (i=1,2,3) \quad (20)$$

$[\mathbf{R}]$ is defined as the inverse of the Jacobian matrix, $[\mathbf{J}^{-1}]$. Then the required partial derivatives are defined as:

$$\frac{\partial N^k}{\partial x_i} = R_{i1} \frac{\partial N^k}{\partial \xi} + R_{i2} \frac{\partial N^k}{\partial \eta} \quad (i=1,2,3) \quad (21)$$

$$\frac{\partial H^k}{\partial x_i} = R_{i3} \frac{\partial H^k}{\partial \zeta} \quad (i=1,2,3) \quad (22)$$

F. STRESS-STRAIN RELATIONSHIP

The strain calculated in Eq. (11) is in the global coordinate, and is transformed to a local coordinate system using

$$\{\varepsilon_{local}\} = [T] \{\varepsilon_{global}\} \quad (23)$$

where $[T]$ is defined in Eq. (10). Stress is calculated from the strain in the local coordinate system using the plane-strain assumption for the in-plane stress components:

$$\begin{aligned} \sigma_x &= \frac{E}{1-\nu^2} (\varepsilon_x + \nu \varepsilon_y) & \tau_{xy} &= G \gamma_{xy} \\ \sigma_y &= \frac{E}{1-\nu^2} (\nu \varepsilon_x + \varepsilon_y) & \tau_{yz} &= K G \gamma_{yz} \\ \sigma_z &= E \varepsilon_z & \tau_{xz} &= K G \gamma_{xz} \end{aligned} \quad (24)$$

where K is the shear correction factor, E is the elastic modulus, G is the shear modulus, and ν is Poisson's ratio. The resulting stresses are in the local coordinate system and are converted to the global coordinate system with

$$\{\sigma_{global}\} = [T]^T \{\sigma_{local}\} \quad (25)$$

$$\{\sigma\} \equiv \{\sigma_x \sigma_y \sigma_z \gamma_{xy} \gamma_{yz} \gamma_{xz}\} \quad (26)$$

where $[T]$ is from Eq. (10).

G. DRILLING MOMENTS

Let u_3 be the transverse deflection as defined in Eq. (6). The work done by a pressure loading, p , on an element is:

$$W = \int_{A^e} u_3 p dA \quad (27)$$

where A^e is the element area. Substituting Eq. (6) into Eq. (27), the work is now:

$$W = \{u_3\}^T \int_{A^e} [N] p dA + \{\theta_3\}^T \int_{A^e} [N^k H^k] p dA \quad (28)$$

The first term gives the conventional forces at each node, while the second term yields the new nodal load, which will be called the Drilling Moment (DM). For example, if p is a concentrated force P at node n , then the drilling moment associated with θ_3^n becomes $\frac{1}{2}\zeta tP$, where t is the shell thickness and the mid-plane is the reference plane.

When P is applied at the top plane, ζ equals one, and the drilling moment is $\frac{1}{2}tP$. If P is applied at the bottom plane (still with a positive loading direction), the drilling moment is $-\frac{1}{2}tP$. That is, the load results in compression or tension in the transverse normal stress depending on whether the loading is applied to the top or bottom surface of the shell. The transverse normal stress is assumed constant through the shell thickness for elastic deformation. However, as plastic deformation progresses, the transverse normal stress becomes non-uniform through the thickness in order to satisfy the yield function.

The drilling moment direction coincides with the direction of drilling rotation, but affects the transverse normal stress. The drilling moment is used as a mathematical convenience, and thus, lacks a physical interpretation.

H. INTERNAL FORCE, MASS AND THEIR ASSEMBLY

The stress resulting from Eq. (25) is then converted to an internal force vector and summed over all integration points using

$$\{f_{int}\} = \int_{-1}^1 \int_{-1}^1 \int_{-1}^1 [\mathbf{B}]^T \{\sigma\} d\xi d\eta d\zeta = \sum_{i=1}^{nx} \sum_{j=1}^{ny} \sum_{k=1}^{nz} [\mathbf{B}]^T \{\sigma\} w_i w_j w_k |\mathbf{J}| \quad (29)$$

where $|\mathbf{J}|$ is the determinant of the Jacobian matrix, nx , ny , and nz are the number of integration points in the ξ , η , and ζ directions, respectively, and w_i , w_j , and w_k are the Gauss weights related to those integration points. The resulting force vector, $\{f_{int}\}$, must have its components transformed back to the global coordinate system using:

$$\{F_{int}^k\}_{(6 \times 1)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & & & \\ 0 & 0 & 0 & [\mathbf{T}_p] & & \\ 0 & 0 & 0 & & & \end{bmatrix} \{f_{int}^k\}_{(6 \times 1)} \quad (k=1,2,3,4) \quad (30)$$

A lumped mass method is used for the element mass matrix. The matrix for each element is diagonal, with equal diagonal elements:

$$[\mathbf{M}_e] = \begin{bmatrix} m_e & 0 & 0 & \dots & 0 \\ 0 & m_e & 0 & \dots & 0 \\ 0 & 0 & m_e & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & m_e \end{bmatrix} \quad (31)$$

where

$$m_e = \frac{\left(\sum_{i=1}^{nx} \sum_{j=1}^{ny} \sum_{k=1}^{nz} w_i w_j w_k |J| \right) \rho}{n} \quad (32)$$

with $n = 4$ in this four-node element. Using a diagonal mass matrix greatly simplifies time integration, since inverting it is a trivial task requiring little computation time. The internal force vector and element mass vector (each 24 by 1) are then assembled into the corresponding system vectors.

I. EXPLICIT TIME INTEGRATION

The use of internal force vectors and explicit time integration negates the need to explicitly form the system stiffness matrices. The acceleration vector is computed from

$$\{\ddot{U}\} = [M]^{-1} \left(\{F_{ext}\}' - \{F_{int}\}' \right) \quad (33)$$

where $\{\ddot{U}\}$ is the system acceleration vector, $[M]$ is the system mass matrix, $\{F_{ext}\}'$ is the system external force vector, and superscript t denotes the time step. Of course, the system mass matrix in Eq. (33) is simply symbolic, since a system mass vector, formed from the diagonal of the mass matrix, is used in actual computation. Velocity and displacement are then found using

$$\begin{aligned} \{\dot{U}\}^{t+\frac{\Delta t}{2}} &= \{\dot{U}\}^{t-\frac{\Delta t}{2}} + \{\ddot{U}\}'^t \Delta t \\ \{U\}^{t+\Delta t} &= \{U\}'^t + \{\dot{U}\}^{t+\frac{\Delta t}{2}} \Delta t \end{aligned} \quad (34)$$

III. DAMAGE CONSTITUTIVE EQUATIONS

A. GURSON'S VOID MODEL

Yielding and plastic deformation in the element follows the model proposed by Gurson for symmetric deformations around a spherical void [17]. The yielding condition is

$$F = \left(\frac{q}{\sigma_0} \right)^2 + 2 q_1 \Phi \cosh \left(- \frac{3}{2} \frac{q_2 p}{\sigma_0} \right) - (1 + q_3 \Phi^2) = 0 \quad (35)$$

where Φ is the current porosity, p is the hydrostatic stress, q is the effective stress, and σ_0 is the current yield stress. This model assumes equivalent yield stress in both tension and compression. The constants q_1 , q_2 , and q_3 were introduced by Tvergaard in order to provide a better match with numerical studies [16]. Aravas provides a detailed explanation of implementing this model in a static finite element algorithm for three-dimensional solid elements [21]. The procedure used here is essentially the same, with some modifications due to the different element formulation. The stress is then transformed to local coordinates, and the internal force vector is computed as described above. One thing to be noted in Eq. (35) is that if Φ is initially 0, the yielding criteria surface is identical to the von Mises yield condition.

After calculating the strain tensor using Eq. (23), any previous plastic strain is subtracted using

$$\{ \varepsilon^e \} = \{ \varepsilon^{total} \} - \{ \varepsilon^p \} \quad (36)$$

The stress is then calculated using Eq. (24) with the components of $\{\varepsilon^e\}$. Using these values, the hydrostatic stress, deviatoric stress and effective stress are calculated as follows:

$$p = \frac{1}{3} (\sigma_{xx} + \sigma_{yy} + \sigma_{zz}) \quad (37)$$

$$\{s\} = \{\sigma\} + p\{\delta\} \quad (38)$$

$$q = \sqrt{\frac{3}{2} (s_1^2 + s_2^2 + s_3^2 + 2(s_4^2 + s_5^2 + s_6^2))} \quad (39)$$

where δ is the Kronecker delta function:

$$\{\delta\} = \{11100\}^T \quad (40)$$

At this point, F is calculated from Eq. (35). If F is greater than zero, indicating plastic flow, iteration is required to determine the new porosity and change in plastic strain. The predictor-corrector method used in Aravas [21] is also used here. Using the values of p and q previously calculated as a first guess, correction factors are calculated using

$$\{C_f\} = [A]^{-1} \{A1\} \quad (41)$$

where

$$\{A1\} = \begin{Bmatrix} -F \\ -\Delta\varepsilon_p \frac{\partial F}{\partial q} - \Delta\varepsilon_q \frac{\partial F}{\partial p} \end{Bmatrix} \quad (42)$$

$$[A] = \begin{bmatrix} K \frac{\partial F}{\partial p} + \frac{\partial F}{\partial \sigma_0} \frac{\partial \sigma_0}{\partial \Delta\varepsilon_p} & -3G \frac{\partial F}{\partial q} + \frac{\partial F}{\partial \sigma_0} \frac{\partial \sigma_0}{\partial \Delta\varepsilon_q} \\ \frac{\partial F}{\partial q} + \Delta\varepsilon_q \left[K \frac{\partial^2 F}{\partial p^2} + \frac{\partial^2 F}{\partial p \partial \sigma_0} \frac{\partial \sigma_0}{\partial \Delta\varepsilon_p} \right] + \Delta\varepsilon_p \frac{\partial^2 F}{\partial q \partial \sigma_0} \frac{\partial \sigma_0}{\partial \Delta\varepsilon_p} & \frac{\partial F}{\partial p} + \Delta\varepsilon_p \left[-3G \frac{\partial^2 F}{\partial q^2} + \frac{\partial^2 F}{\partial q \partial \sigma_0} \frac{\partial \sigma_0}{\partial \Delta\varepsilon_q} \right] + \Delta\varepsilon_q \frac{\partial^2 F}{\partial q \partial \sigma_0} \frac{\partial \sigma_0}{\partial \Delta\varepsilon_q} \end{bmatrix} \quad (43)$$

$$\{C_f\} = \begin{Bmatrix} \alpha \\ \beta \end{Bmatrix} \quad (44)$$

and

$$\begin{aligned}
\frac{\partial F}{\partial q} &= \frac{2q}{\sigma_0^2} \\
\frac{\partial^2 F}{\partial q^2} &= \frac{2}{\sigma_0^2} \\
\frac{\partial F}{\partial p} &= 2q_1 \Phi \left(\frac{-3q_2}{2\sigma_0} \right) \sinh \left(\frac{-3q_2 p}{2\sigma_0} \right) \\
\frac{\partial^2 F}{\partial p^2} &= 2q_1 \Phi \left(\frac{3q_2}{2\sigma_0} \right)^2 \cosh \left(\frac{-3q_2 p}{2\sigma_0} \right) \\
\frac{\partial F}{\partial \sigma_0} &= \frac{-q^2}{2\sigma_0^3} + 2q_1 \Phi \frac{3q_2 p}{2\sigma_0^2} \sinh \left(\frac{-3q_2 p}{2\sigma_0} \right) \\
\frac{\partial^2 F}{\partial p \partial \sigma_0} &= -2q_1 \Phi p \left(\frac{3q_2}{2\sigma_0^2} \right)^2 \cosh \left(\frac{-3q_2 p}{2\sigma_0} \right) + \frac{3q_2}{2\sigma_0^3} \sinh \left(\frac{-3q_2 p}{2\sigma_0} \right) \\
\frac{\partial^2 F}{\partial q \partial \sigma_0} &= \frac{-q}{\sigma_0^3}
\end{aligned} \tag{45}$$

The values for α and β are used to correct the change in strain caused by hydrostatic pressure and the change in strain caused by effective stress (See Eq. (46)), which are then used to determine the change in the plastic strain vector, as shown in Eq. (47).

$$\begin{aligned}
\Delta \varepsilon_p^{new} &= \Delta \varepsilon_p^{old} + \alpha \quad (\Delta \varepsilon_p^0 = 0) \\
\Delta \varepsilon_q^{new} &= \Delta \varepsilon_q^{old} + \beta \quad (\Delta \varepsilon_q^0 = 0)
\end{aligned} \tag{46}$$

$$\{ \Delta \varepsilon^p \} = \frac{1}{3} \Delta \varepsilon_p \{ \delta \} + \Delta \varepsilon_q \left(\frac{3}{2q} \right) \{ s \} \tag{47}$$

This change in plastic strain, $\{ \Delta \varepsilon^p \}$, is then added to the total plastic strain, and the new elastic strain is calculated using Eq. (36), and the stress vector is calculated again using Eq. (24). Next, the change in void content, or porosity, is calculated as:

$$\Delta \Phi = \Delta \Phi_{growth} + \Delta \Phi_{nucleation} \tag{48}$$

$$\Delta \Phi_{growth} = (1 - \Phi) \sum \varepsilon_{ii}^p \tag{49}$$

$$\Delta \Phi_{nucleation} = \frac{\Phi_N}{s_N \sqrt{2\pi}} \exp \left[-\frac{1}{2} \left(\frac{\epsilon^p - \epsilon_N}{s_N} \right)^2 \right] \Delta \epsilon_p \quad (50)$$

where Φ_N is the volume fraction of void nucleating particles and ϵ_N and s_N are the mean and standard deviation of a normal distribution of nucleation strain, as suggested by Chu and Needleman [15] and utilized by Aravas [21]. Then the effective plastic strain, and void content are updated with

$$\Delta \epsilon^p = \frac{-p \Delta \epsilon_p + q \Delta \epsilon_q}{(1 - \Phi) \sigma_0} \quad (51)$$

$$\epsilon_{i+\Delta t}^p = \epsilon_i^p + \Delta \epsilon^p \quad (52)$$

where ϵ_i^p is the effective plastic strain from the previous time step. At this point, the change in yield stress due to strain hardening is calculated (see below). If either α or β is greater than a predetermined tolerance, the process iterates beginning with Eq. (41). The tolerance used for all examples presented in the paper is 1.0×10^{-4} .

B. IMPROVING STABILITY IN THE CONSTITUTIVE EQUATIONS

Stability in the procedure outlined above is very dependent on the order in which the various equations are evaluated. Aravas discusses the stability problem when considering problems involving large plastic strains [21]. While the change in plastic strain from one time step to the next will theoretically be small, high strain rate and changes in the strain-hardening characteristics of the material act to degrade stability. Equations (42), (43) and (45) are not complete in that they do not contain all of the derivative terms required by the chain rule.

The predictor-corrector method used is based on Newton's method, where the correction factors, α and β , are found from

$$\begin{bmatrix} f_{1,\Delta\epsilon_p} & f_{1,\Delta\epsilon_q} \\ f_{2,\Delta\epsilon_p} & f_{2,\Delta\epsilon_q} \end{bmatrix} \begin{Bmatrix} \alpha \\ \beta \end{Bmatrix} = \begin{Bmatrix} f_1 \\ f_2 \end{Bmatrix} \quad (53)$$

where

$$f_1 = \left(\frac{q}{\sigma_0} \right)^2 + 2q_1\Phi \cosh\left(\frac{-3q_2p}{2\sigma_0} \right) - (1 + q_3\Phi^2) \quad (54)$$

$$f_2 = \Delta\epsilon_p \frac{\partial f_1}{\partial q} + \Delta\epsilon_q \frac{\partial f_1}{\partial p} \quad (55)$$

$$f_{1,\Delta\epsilon_p} = K \frac{\partial f_1}{\partial p} + \frac{\partial f_1}{\partial \Phi} \frac{\partial \Phi}{\partial \Delta\epsilon_p} + \frac{\partial f_1}{\partial \sigma_0} \frac{\partial \sigma_0}{\partial \Delta\epsilon_p} \quad (56)$$

$$f_{1,\Delta\epsilon_q} = -3G \frac{\partial f_1}{\partial q} + \frac{\partial f_1}{\partial \Phi} \frac{\partial \Phi}{\partial \Delta\epsilon_q} + \frac{\partial f_1}{\partial \sigma_0} \frac{\partial \sigma_0}{\partial \Delta\epsilon_q} \quad (57)$$

$$f_{2,\Delta\epsilon_p} = \frac{\partial f_1}{\partial q} + \Delta\epsilon_p \frac{\partial^2 f_1}{\partial \sigma_0 \partial q} \frac{\partial \sigma_0}{\partial \Delta\epsilon_p} + \Delta\epsilon_q \left[K \frac{\partial^2 f_1}{\partial p^2} + \frac{\partial^2 f_1}{\partial p \partial \Phi} \frac{\partial \Phi}{\partial \Delta\epsilon_p} + \frac{\partial^2 f_1}{\partial p \partial \sigma_0} \frac{\partial \sigma_0}{\partial \Delta\epsilon_p} \right] \quad (58)$$

$$f_{2,\Delta\epsilon_q} = \frac{\partial f_1}{\partial p} + \Delta\epsilon_q \left[\frac{\partial^2 f_1}{\partial p \partial \Phi} \frac{\partial \Phi}{\partial \Delta\epsilon_q} + \frac{\partial^2 f_1}{\partial p \partial \sigma_0} \frac{\partial \sigma_0}{\partial \Delta\epsilon_q} \right] + \Delta\epsilon_p \left[-3G \frac{\partial^2 f_1}{\partial q^2} + \frac{\partial^2 f_1}{\partial q \partial \sigma_0} \frac{\partial \sigma_0}{\partial \Delta\epsilon_q} \right] \quad (59)$$

Derivative terms that are zero have been dropped, and K and G are the bulk and shear moduli, respectively. f_1 is Gurson's void function, and f_2 is the flow rule, both of which are driven to zero. As void content increases, the number of iterations required to achieve convergence increases dramatically, and often diverges, instead. By removing all partial derivatives relating to void content, the stability of the algorithm is greatly increased, at the cost of only 2 to 3 extra iterations, depending on the current void content. The dependence

on the current yield stress is retained to allow convergence across a transition in the slope of the yield stress versus strain relationship.

The partial derivatives of yield stress with respect to $\Delta\epsilon_p$ and $\Delta\epsilon_q$ are computed by calculating the slope from the current yield stress to the yield stress corresponding to the increase in plastic strain from the corrected values of $\Delta\epsilon_p$ and $\Delta\epsilon_q$. This means that these derivatives will be zero on the first iteration, since both of the control variables are initialized to zero. Several error traps must be included to prevent round-off and truncation error from causing the algorithm to diverge, and to prevent porosity from decreasing or becoming negative. One of the assumptions used in this implementation is that once voids form, they do not disappear. In other words, voids do not disappear when the element is placed in compression.

C. STRAIN HARDENING

Modeling the nonlinear elasto-plastic behavior of the material used is simplified by constructing a piece-wise linear version of the stress-strain plot. Using the tangent modulus, E_T , for each piece-wise region, the yield stress is calculated with

$$\sigma_0^{t+\Delta t} = \sigma_0^0 + \sum_{i=1}^{j-1} E_{Ti} (\epsilon_i - \epsilon_{i-1}) + E_{Tj} (\epsilon_i^{eff} - \epsilon_{j-1}) \quad (60)$$

where $\sigma_0^{t+\Delta t}$ is the yield stress for this time step, σ_0^0 is the original yield stress, ϵ_i^{eff} is the total effective strain for the time step under consideration, and ϵ_i is the upper strain limit of the i th linear segment. ϵ_0 is the original yield strain, and is calculated by the program as

simply σ_0^0/E . The current effective elastic strain is calculated by dividing the current yield stress by the elastic modulus, E . This is added to the current effective plastic strain to obtain the total effective strain.

As an example, let the total effective strain from Eq. (49) lie within the second work-hardening segment. The new yield stress is

$$\sigma_0^{t+\Delta t} = \sigma_0^0 + E_{T1}(\varepsilon_1 - \varepsilon_0) + E_{T2}(\varepsilon_{t+\Delta t}^{eff} - \varepsilon_1) \quad (61)$$

Figure 3 illustrates this example. The algorithm calculates the new yield stress as a function of the cumulative effective plastic strain, the current effective elastic strain, and the original yield stress for each iteration of the damage constitutive equations.

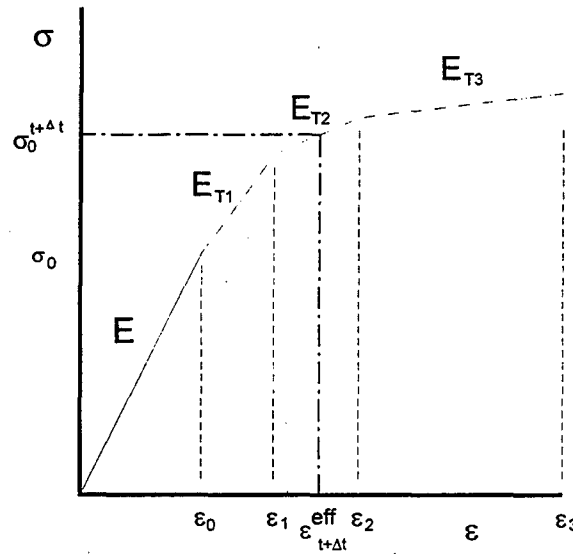


Figure 3. Calculating a New Yield Stress.

IV. HOURGLASS MODE CONTROL

Only one integration point is used in each plane parallel to the reference plane, which results in under-integration in the ξ - η plane. This leads to spurious, hourglass, or zero-energy modes in the element, which will yield useless results if left uncontrolled. The effects of these modes are shown in Fig. 4, a pinched cylinder where one eighth of the structure is modeled by utilizing symmetry boundary conditions. Clearly, no useful information can be obtained from the resulting solution.

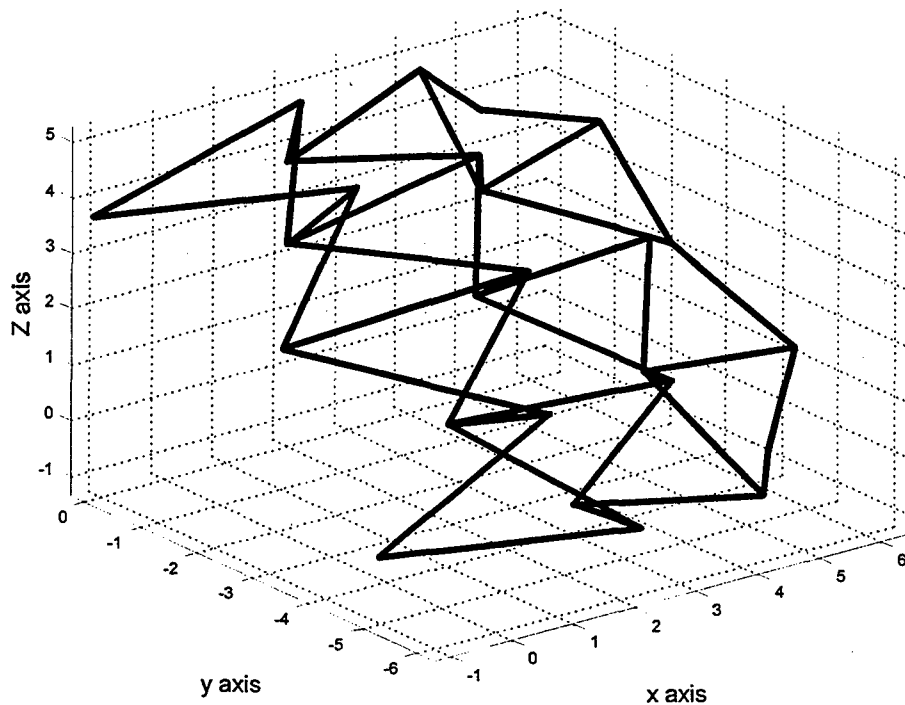


Figure 4. Hourglass Modes in a Pinched Cylinder Model.

Belytschko, et al., proposed an efficient means of controlling the hourglass modes of a similar element [25]. The method described uses a portion of a stiffness matrix generated by full integration in all directions to modify the stiffness matrix generated by under-integration. Although the formulation proposed in this paper does not use a stiffness matrix, a similar approach is just as effective in controlling these modes.

Rather than fully integrating in all three directions, the element is fully integrated in the ξ - η plane, but under-integrated in the ζ direction, as shown in Fig. 5. The procedure described above for calculating the internal force vector is followed to generate an internal force vector related to these four integration points.

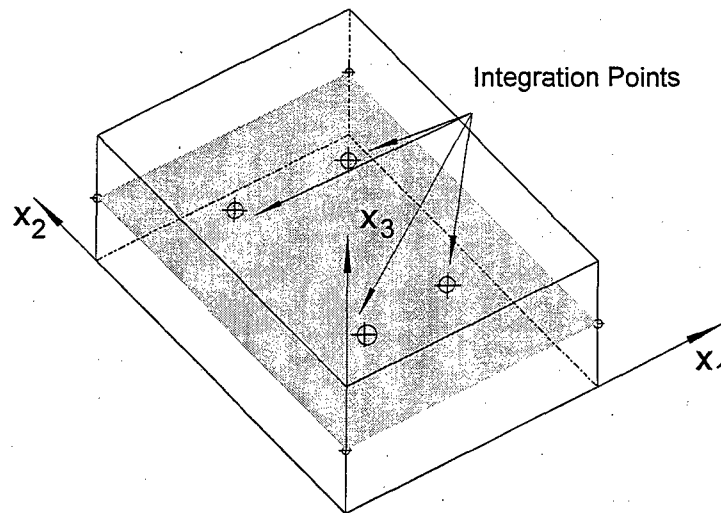


Figure 5. Hourglass Mode Control Integration Points.

The algorithm for calculating strain, stress, and force for the hourglass mode control integration points is identical to the algorithm used to produce the main internal force vector,

with the exception of plastic strain. The damage constitutive equations described in the previous section are not utilized for hourglass mode control. The internal forces generated from the two integration schemes are treated like the stiffness matrices in Belytschko et al. [25]. For nz integration points through the element thickness, the new force vector is

$$\{f_{int}\} = \{f_{int}^{1 \times 1 \times nz}\} + h \{f_{hour}\} \quad (62)$$

where

$$\{f_{hour}\} = \{f_{int}^{2 \times 2 \times 1}\} - \{f_{int}^{1 \times 1 \times nz}\} \quad (63)$$

and

$$h = \frac{r t^2}{A} \quad (64)$$

The variable h is used here in Eqs. (62) and (64) instead of the ε used in Belytschko et al. [25], to avoid confusion with the various strains discussed in this paper. The variables used to calculate h are the element thickness (t) and the element's surface area (A). The effect of r follows that described in Belytschko et al. [25], and is set to 0.05. The range of values for r that effectively controls the hourglass modes, but does not greatly affect the overall element stiffness is roughly 0.046 to 0.057 (determined experimentally). Since the elements are in arbitrary orientation in 3-D space, the area calculation is computed as the sum of the area of the two triangles formed by dividing the element at the diagonal between nodes one and three:

$$A = \frac{1}{2} \sqrt{l_1^2 l_2^2 - D_1^2 + l_3^2 l_4^2 - D_2^2} \quad (65)$$

where

$$D_1 = (x_1^1 - x_1^2)(x_1^3 - x_1^2) + (x_2^1 - x_2^2)(x_2^3 - x_2^2) + (x_3^1 - x_3^2)(x_3^3 - x_3^2) \quad (66)$$

$$D_2 = (x_1^1 - x_1^4)(x_1^3 - x_1^4) + (x_2^1 - x_2^4)(x_2^3 - x_2^4) + (x_3^1 - x_3^4)(x_3^3 - x_3^4)$$

and

$$\begin{aligned} l_1 &= \sqrt{(x_1^2 - x_1^1)^2 + (x_2^2 - x_2^1)^2 + (x_3^2 - x_3^1)^2} \\ l_2 &= \sqrt{(x_1^3 - x_1^1)^2 + (x_2^3 - x_2^1)^2 + (x_3^3 - x_3^1)^2} \\ l_3 &= \sqrt{(x_1^4 - x_1^1)^2 + (x_2^4 - x_2^1)^2 + (x_3^4 - x_3^1)^2} \\ l_4 &= \sqrt{(x_1^1 - x_1^4)^2 + (x_2^1 - x_2^4)^2 + (x_3^1 - x_3^4)^2} \end{aligned} \quad (67)$$

and (x_1^k, x_2^k, x_3^k) is the location of node k in the global coordinate system. For these calculations, the element is assumed to be flat (no curvature along either the ξ or η directions). The effectiveness of this method of control is shown in Fig. 6, the same problem as shown in Fig. 4, but with the hourglass mode control described above.

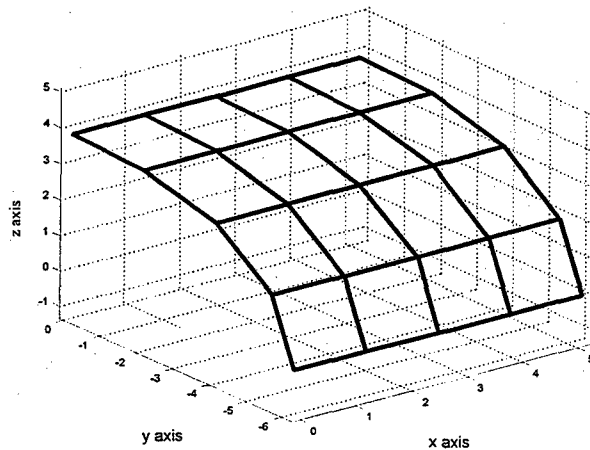


Figure 6. Pinched Cylinder Model with Hourglass Mode Control.

All verification problems analyzed in the following section were completed with hourglass control enabled. The current implementation uses hourglass control consistently, but allows easy modification to make hourglass control a user-defined option. The internal hourglass mode control can be disabled when the element is used in a general finite element program that includes various methods of spurious mode control.

VI. NUMERICAL EXAMPLES

The element presented here has been extensively tested using a variety of problems in which an analytic solution was available, and has produced satisfactory results in all cases studied. The transformation matrices and constitutive equations were verified with specifically tailored problems, and the element passes the patch test. The examples presented below are representative of the test cases used to verify the element, and are presented in the order of curvature: flat, singly curved, and doubly curved. For each case, an elastic solution is compared to available results, and then the elasto-plastic solutions are presented. Examples A through I were solved using a locally written, research oriented finite element program in conjunction with a locally developed preprocessor and postprocessor.

A. ELASTIC PLATE

A plate clamped on all four sides is subjected to a concentrated force at its center. The elastic modulus is 10 Msi (68.95 GPa), the density is 0.1647 slugs/in³ (0.147 kg/cm³), and Poisson's ratio is 0.2. The dimensions of the plate are 10 in x 10 in x 0.1 in thick (25.4cm x 25.4 cm x 0.254 cm). The yield stress is set high enough to ensure a completely elastic response. Two integration points through the thickness are used. The applied force is 40 lbf (177.9 N). The finite element mesh uses symmetry to model one quarter of the plate, with appropriate symmetry boundary conditions applied. Both a 2x2 (4 element) and 4x4 (16 element) mesh are used in the finite element analysis. The results for both meshes

and the analytic solution are shown in Table 1. Using a four-element mesh, the new shell element obtained a displacement within 3.27% of the analytic solution, and 0.82% using a 16 element mesh.

Table 1. Comparison of Results for Elastic Clamped Plate.

Analysis Type	Center Node Peak Displacement (in)
2 x 2 FE Mesh (Dynamic)	-4.74×10^{-2}
4 x 4 FE Mesh (Dynamic)	-4.94×10^{-2}
Analytic (Twice the Static Solution)	-4.90×10^{-2}

B. THICK CLAMPED PLATE UNDER PRESSURE LOAD IN ELASTO-PLASTIC REGION

A thick steel plate, clamped on all four sides, is subjected to dynamic pressure load.

The plate is 6m by 6m by 0.6m thick (for a 10:1 ratio). Table 2 shows the material properties for the structure. Table 3 shows the properties for the void model.

Table 2. Material Properties of Clamped Plate.

Property	Value	Units
Elastic Modulus (E)	2×10^{11}	Pa
Tangent Modulus (E_T)	2×10^{10}	Pa
Density (ρ)	7850	kg/m ³
Poisson's ratio (ν)	0.29	(none)
Yield Stress (S_{yp})	2.5×10^8	Pa

Table 3. Void Characteristics of Clamped Plate.

Initial Void Content (Φ_0)	0.0
Nucleating Particle Content (Φ_N)	0.04
Mean Nucleation Strain (e_N)	0.3
Nucleation Strain Standard Deviation (s_N)	0.1
Model Constant q_1	1.5
Model Constant q_2	1.0
Model Constant q_3 ($= q_1^2$)	2.25

One quarter of the plate is modeled with a three by three element mesh, for a total of nine elements, with appropriate symmetry boundary conditions applied, and is shown below in Fig. 8.

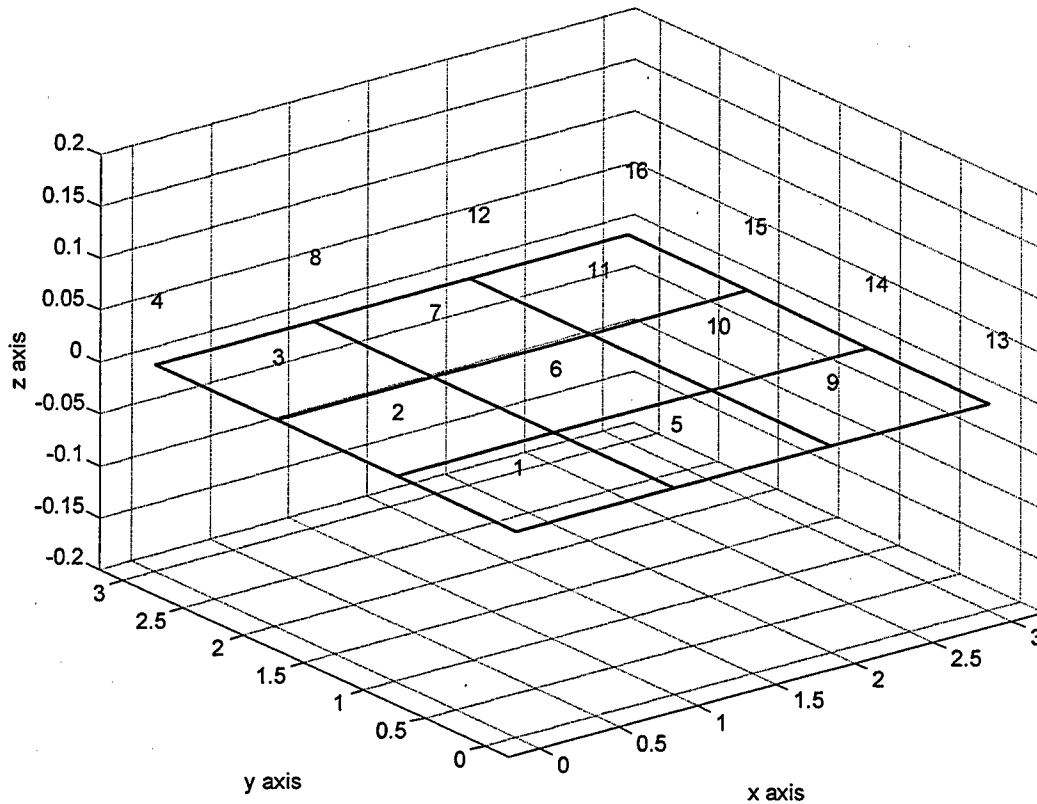


Figure 7. Nine Element Clamped Plate Mesh.

Four integration points are used through the thickness of each element. The calculation time step is 10^{-5} seconds, while the data is plotted at 2×10^{-4} second increments. The analysis terminates at 0.04 seconds. The plate is subjected to a uniformly distributed pressure acting downwards, and includes the appropriate drilling moment. The pressure

increases linearly from 0.0 Pa at the start to 80 MPa at 0.01 seconds, then remains constant for the duration of the analysis. Three cases were analyzed: no void effects, void growth effects only, and void growth and nucleation. Figure 9 illustrates the effect of voids on the effective stress versus the effective strain relationship in the top of one of the border elements. When void effects are not included, the Von-Mises Equivalent (VME) stress follows the yield stress in the plastic region (See Fig. 9a).

Including void effects causes the yielding before the VME stress reaches the yield stress. As the void content increases with continued plastic flow, the VME stress falls farther below the yield stress, as shown in Fig. 9b (see Eq. (35)). Table 4 summarizes the results of the three cases. The most significant difference is in the transverse normal stress, σ_{zz} . This difference resulted in a 2.0% increase in the effective plastic strain, and a 2.6% increase in the peak deflection of the center of the plate.

Another interesting point is that the transverse normal stress was compressive and constant throughout the thickness up until plastic flow, as assumed in the formulation, then varied as the stress in the bottom fiber decreased and became tensile. Figure 10 shows the variation of transverse normal stress through the thickness of the element at the time of peak stress. Eventually, the transverse normal stress varied from compressive at the top, where the pressure was applied, and tensile at the bottom. The examples that follow will not include a separate analysis for void growth effects only, since the difference of including nucleation effects at the resulting small plastic strains obtained do not cause significantly different results.

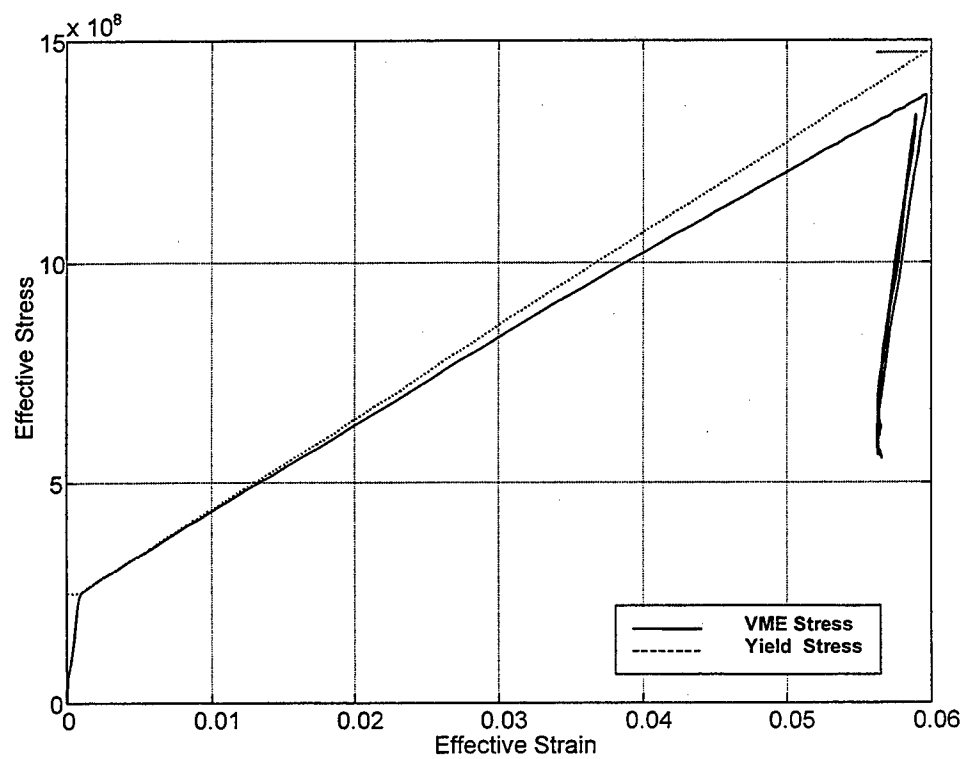
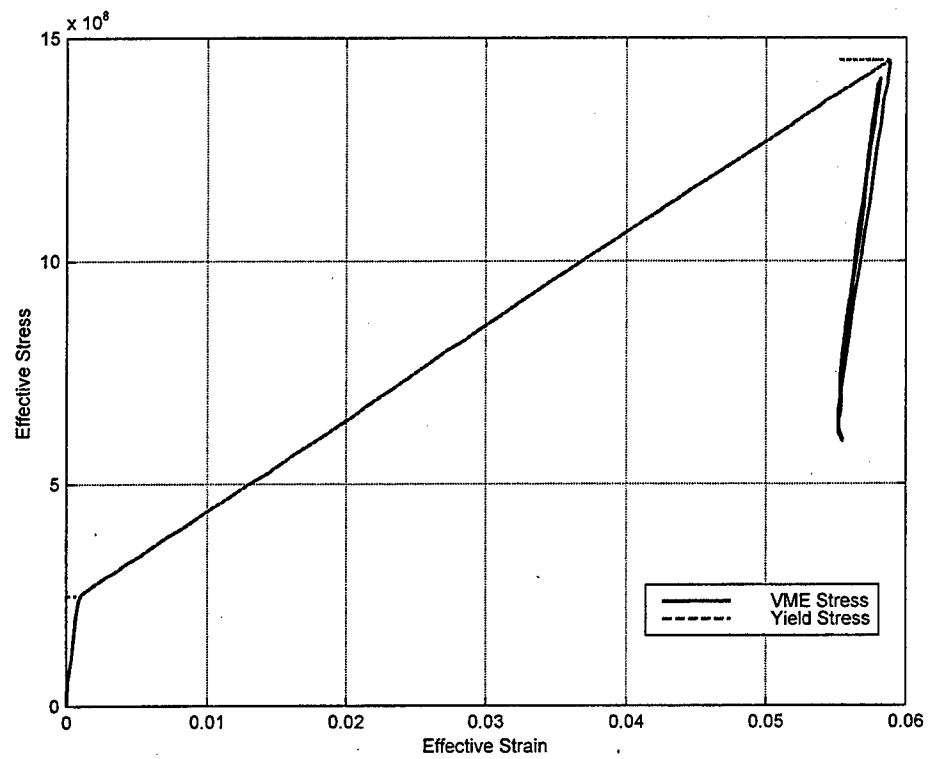


Figure 8. Void Effects in a Clamped Plate with Pressure Loading: Top (a) - No Void Effects, Bottom (b) - Void Effects.

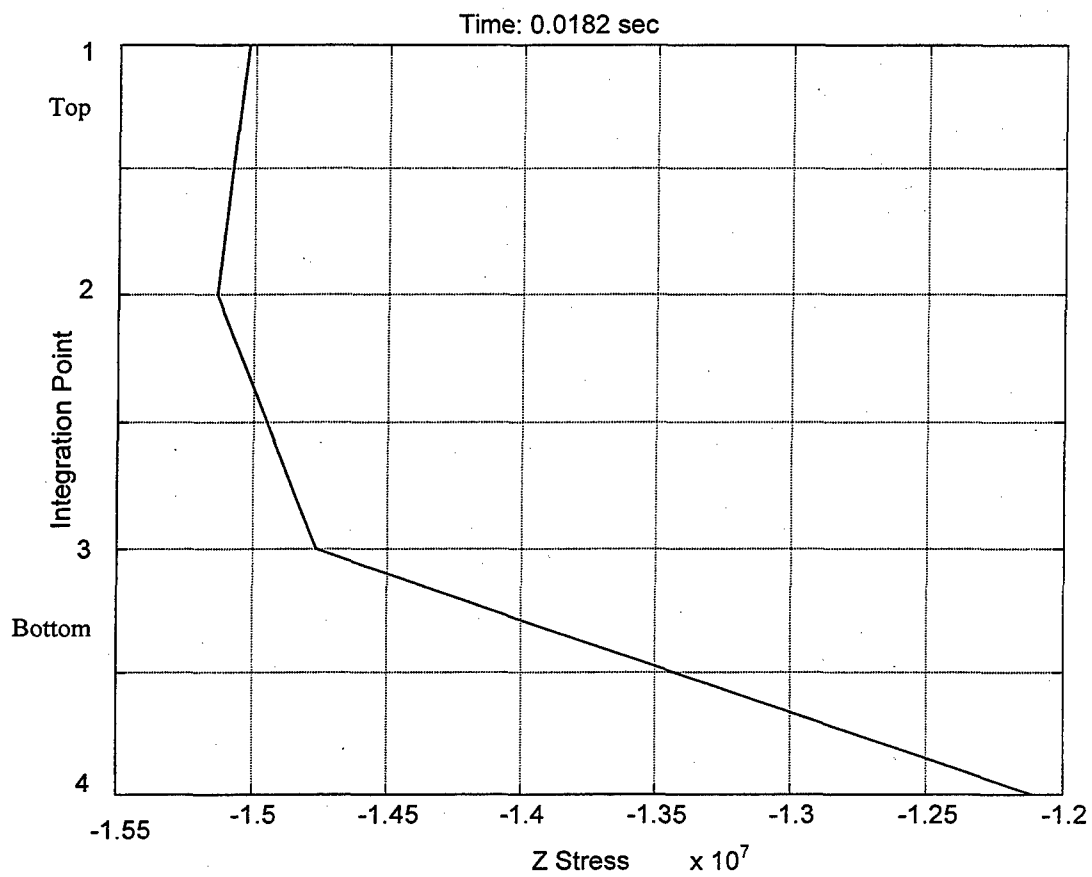


Figure 9. Transverse Normal Stress Variation through Shell Thickness: Clamped Plate with Pressure Loading and Void Effects.

Table 4. Summary of Results for Clamped Plate with Pressure Load.

Peak Values for Element #3	No Void Effects	Void Growth	Growth and Nucleation
σ_{VM} (GPa)	1.4498	1.3789	1.3784
$\epsilon_{effective}$	0.0588	0.0597	0.0597
σ_{xx} (GPa)	1.4501	1.3603	1.3598
σ_{yy} (GPa)	0.6561	0.6030	0.6073
σ_{zz} (GPa) (Max Tension)	-0.0003	0.0065	0.0091
σ_{zz} (GPa) (Max Compression)	-0.0363	-0.0322	-0.0320
$\epsilon^{plastic}$ (effective)	0.0540	0.0551	0.0551
Φ (Porosity)	NA	0.0355	0.0357
Deflection (m) (Center Node)	-0.3801	-0.3897	-0.3898

C. THICK CLAMPED PLATE WITH CENTRAL POINT LOAD IN THE ELASTO-PLASTIC REGION

The geometry and material properties of this example are identical to those used in the previous example. The pressure load has been replaced with a point load at the center node. Four cases are studied: without void effects or a drilling moment, with void effects, with a drilling moment applied, and with both void effects and a drilling moment applied.

All four sides are clamped, and the center node displacement is restricted in the x and y directions. Then, a DM equal to the applied force times one-half the plate thickness is applied for both the no-void and void cases. The time history of the stress components in the center element is shown in Fig. 10, and the relationship between porosity and effective plastic strain is illustrated in Fig. 11. The results for all four cases are summarized in Table

5.

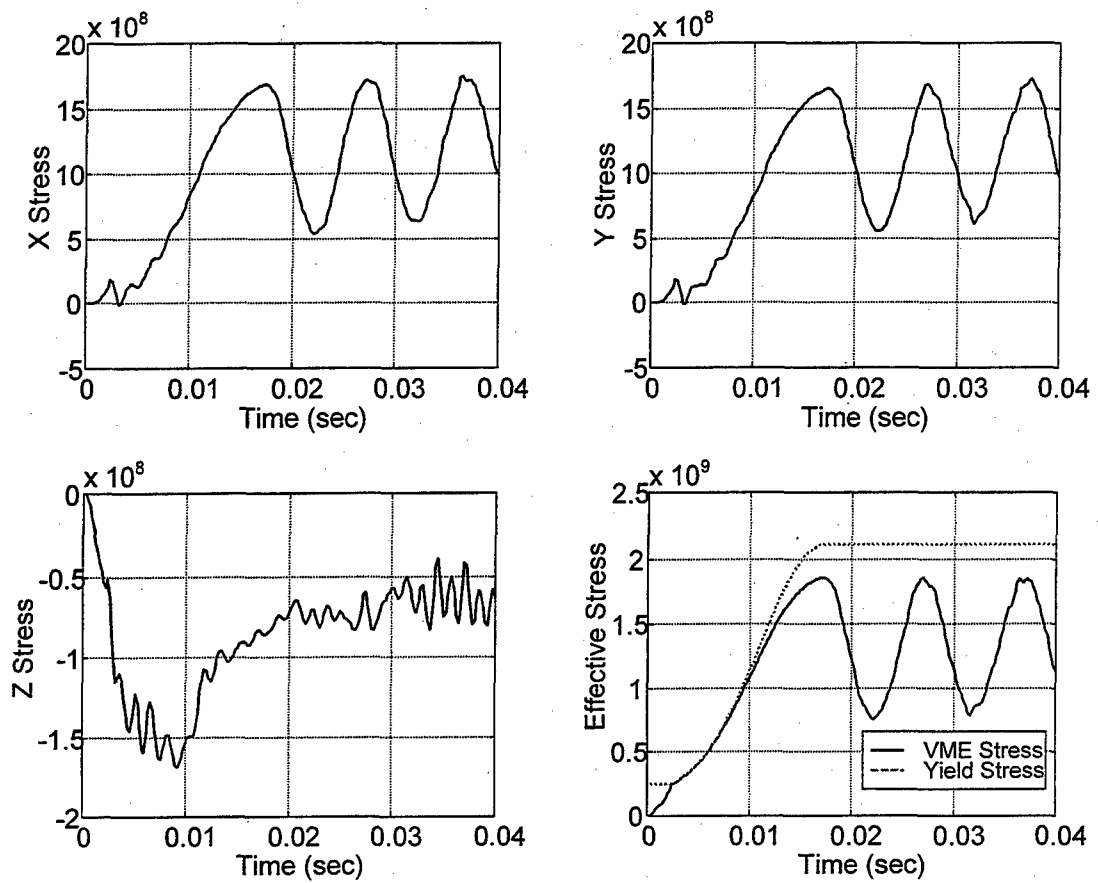


Figure 10. Stress Component Time History in Bottom Fiber: Clamped Plate with Concentrated Load, Void Effects, and Drilling Moment Applied.

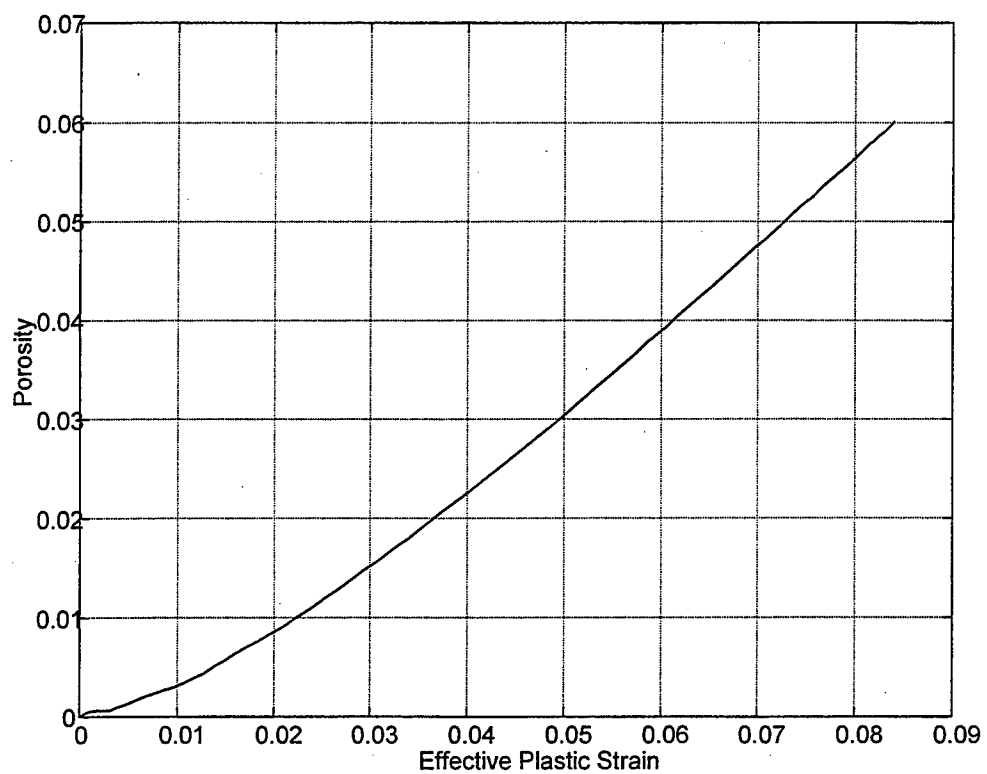


Figure 11. Porosity versus Effective Plastic Strain in Bottom Fiber: Clamped Plate with Concentrated Load, Void Effects, and Drilling Moment Applied.

Table 5. Summary of Results for Clamped Plate Subjected to Point Force.

Peak Values for Center Element	No Voids No DM	Voids No DM	No Voids Drilling Moment	Voids Drilling Moment
σ_{VM} (GPa)	2.1119	1.9021	2.0540	1.8590
$\epsilon_{effective}$	0.0892	0.0929	0.0865	0.0887
σ_{xx} (GPa)	2.0008	1.8407	2.0000	1.7498
σ_{yy} (GPa)	1.9588	1.7245	1.8589	1.7234
σ_{zz} (GPa) (Max Tension)	0.0048	0.0213	-0.0003	-0.0003
σ_{zz} (GPa) (Max Compression)	-0.0036	-0.0057	-0.1751	-0.1691
$\epsilon^{plastic}$ (effective)	0.0838	0.0880	0.0812	0.0840
Φ (Porosity)	NA	0.0663	NA	0.0601
Deflection (m) (Center Node)	-0.4413	-0.4544	-0.4350	-0.4451

The drilling moment increases the effective stress on the fiber in compression, and decreases the effective stress on the fiber in tension. The reduction in the tensile stress results in a reduction in the effective plastic strain. Including void effects decreases the VME stress on the fiber in tension, and slightly increases the VME stress on the fiber in compression. The effective plastic strain is also increased. All of these results indicate that this formulation correctly predicts, in a qualitative sense, the effects of drilling moments and void growth and nucleation. Including only void effects increased the effective plastic strain by 5.0%, while including only the drilling moment decreased the effective plastic strain by 3.1%. Including both void effects and the drilling moment increased the effective plastic strain by 0.2%, indicating the importance of applying both effects together.

D. SIMPLY SUPPORTED PLATE WITH CENTRAL POINT LOAD IN THE ELASTO-PLASTIC REGION

The material properties, geometry, mesh, and time values from the clamped plate

problem above are used here. The magnitude of the force is 8×10^8 N, and the drilling moment, when applied, is -2.4×10^8 N. The same four cases are analyzed: no void or drilling moment effects, drilling moment effects only, void effects only, and both void and drilling moment effects. The analysis results for all four cases are summarized in Table 6.

Table 6. Summary of Results for a Simply Supported Plate with a Point Force.

Peak Values for Center Element	No Voids No DM	No Voids Drilling Moment	Voids No DM	Voids and Drilling Moment
σ_{VM} (GPa)	3.0874	3.0089	2.5061	2.4881
$\epsilon_{\text{effective}}$	0.1360	0.1321	0.1450	0.1393
σ_{xx} (GPa)	3.0253	2.9123	2.4686	2.4206
σ_{yy} (GPa)	3.0459	2.8889	2.4612	2.3928
σ_{zz} (GPa) (Max Tension)	0.0060	-0.0003	0.0114	-0.0003
σ_{zz} (GPa) (Max Compression)	-0.0035	-0.1802	-0.0035	-0.1722
$\epsilon^{\text{plastic}}$ (effective)	0.1277	0.1242	0.1382	0.1327
Φ (Porosity)	NA	NA	0.1197	0.1092
Deflection (m) (Center Node)	-0.9219	-0.8990	-0.9770	-0.9420

The qualitative results for this example are the same as for the clamped plate. Since the applied force causes greater initial yielding, void growth and nucleation has a greater effect. The drilling moment, when added to the void effects, reduces the amount of effective plastic strain in tension and displacement.

E. ELASTIC PINCHED CYLINDER

An open-ended cylinder of radius 5.0 in., length 10.35 in., and thickness 0.094 in. is subjected to a pinching load of 100 lbf (See Fig. 6). The elastic modulus is 10.5 msi, Poisson's ratio is 0.3125, and the density is 3.125×10^{-3} slugs/in². The load is applied as a step function beginning at time $t = 0$ seconds. Using symmetry, the problem was reduced to a one-eighth section of the cylinder. The dynamic value should be twice the analytic static value. Inextensional shell theory gives a static radial contraction of 0.1117 in. The maximum radial contraction of the model is 0.1995 in., which translates to a static radial contraction of 0.09975 in. Using a 256-element mesh (16 by 16), the maximum radial contraction was 0.2207 in., for a static contraction of 0.1104 in. The results are summarized in Table 7. The 16-element solution is within 10.7% of the analytic solution, while the 256-element mesh is within 1.21%.

Table 7. Comparison of Results for Elastic Pinched Cylinder.

Analysis Type	Radial Contraction (in)
Finite Element with 16 Element Mesh	0.1995
Finite Element with 256 Element Mesh	0.2207
Analytic (Twice the Static Solution)	0.2234

F. THICK PINCHED CYLINDER IN THE ELASTO-PLASTIC REGION

The material and void properties shown in Tables 2 and 3 are used for an open-ended cylinder with a pinching force at its center. The top half of the cylinder is modeled using a

36 element mesh with appropriate symmetry boundary conditions along the bottom edge of mesh, as shown in Fig. 13. The analysis is calculated in 10^{-5} second steps, with output every 2×10^{-4} seconds, from 0.0 seconds to 0.04 seconds. The pinching force is 66.792 kN on each side, with a drilling moment of 848.258 Nm applied on the cases indicated. The same four cases are compared: no voids or drilling moment, void effects only, drilling moment only, and both void effects and drilling moment (DM). The effects of void growth and nucleation and the drilling moment is shown in the stress component time histories; Fig. 14 shows the stresses without voids or a drilling moment, and Fig. 15 shows the stresses with both effects included. The results for all cases are summarized in Table 8.

Table 8. Summary of Results for Elasto-Plastic Pinched Cylinder.

Peak Values for Center Element	No Voids No DM	No Voids Drilling Moment	Voids No DM	Voids and Drilling Moment
σ_{VM} (MPa)	262.40	265.05	259.28	283.49
$\epsilon_{effective}$ ($\times 10^3$)	1.6504	1.5722	1.4273	2.7049
σ_{xx} (MPa)	182.46	160.38	170.12	150.31
σ_{yy} (MPa)	291.71	283.77	291.83	297.97
σ_{zz} (MPa) (Max Tension)	5.0619	-0.0238	1.0270	7.5235
σ_{zz} (MPa) (Max Compression)	-7.6358	-28.739	-7.0364	-35.373
$\epsilon^{plastic}$ (effective) ($\times 10^3$)	0.7828	0.7583	0.5771	1.6923
Φ (Porosity) ($\times 10^3$)	NA	NA	0.4773	1.1970
Deflection (mm) (Center Node) Global Maximum	2.7589	12.919	-0.0220	-0.0214
Deflection (mm) (Center Node) Global Minimum	-31.750	-31.750	-31.750	-31.750

Since there is only a small amount of plastic strain, the effects of void growth and nucleation are greatly reduced. Even with this small amount of plastic flow, the effective plastic strain was increased by over 116 percent with the inclusion of both drilling moment and void effects, while the peak center node displacement was only increased by approximately 4.5 percent.

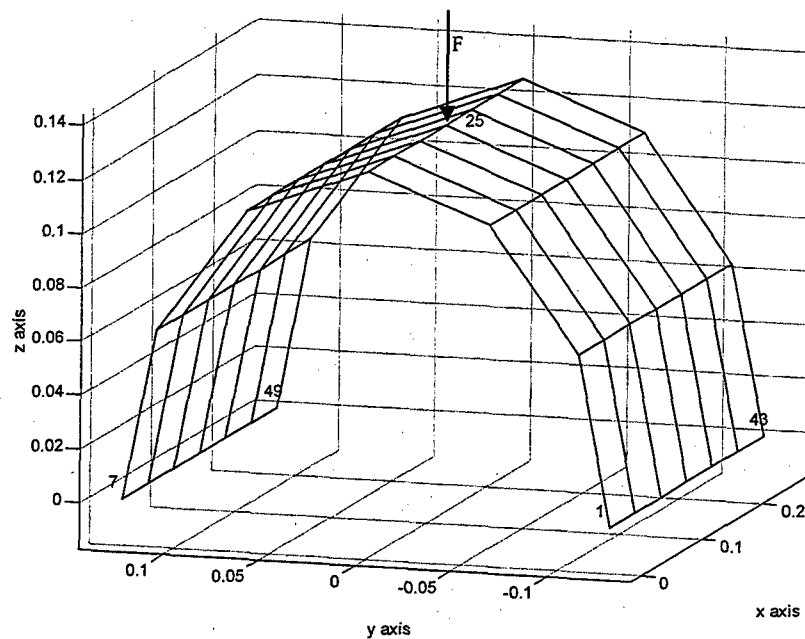


Figure 12. Mesh Structure for Pinched Cylinder.

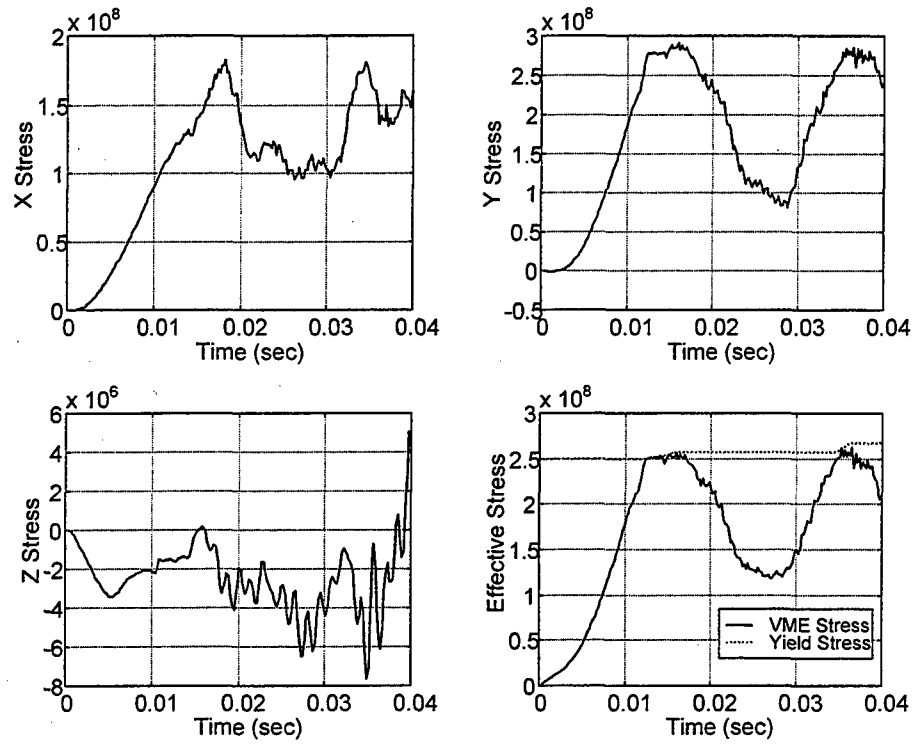


Figure 13. Stress Component Time History in Bottom Fiber: Pinched Cylinder, No Void Effects or Drilling Moment.

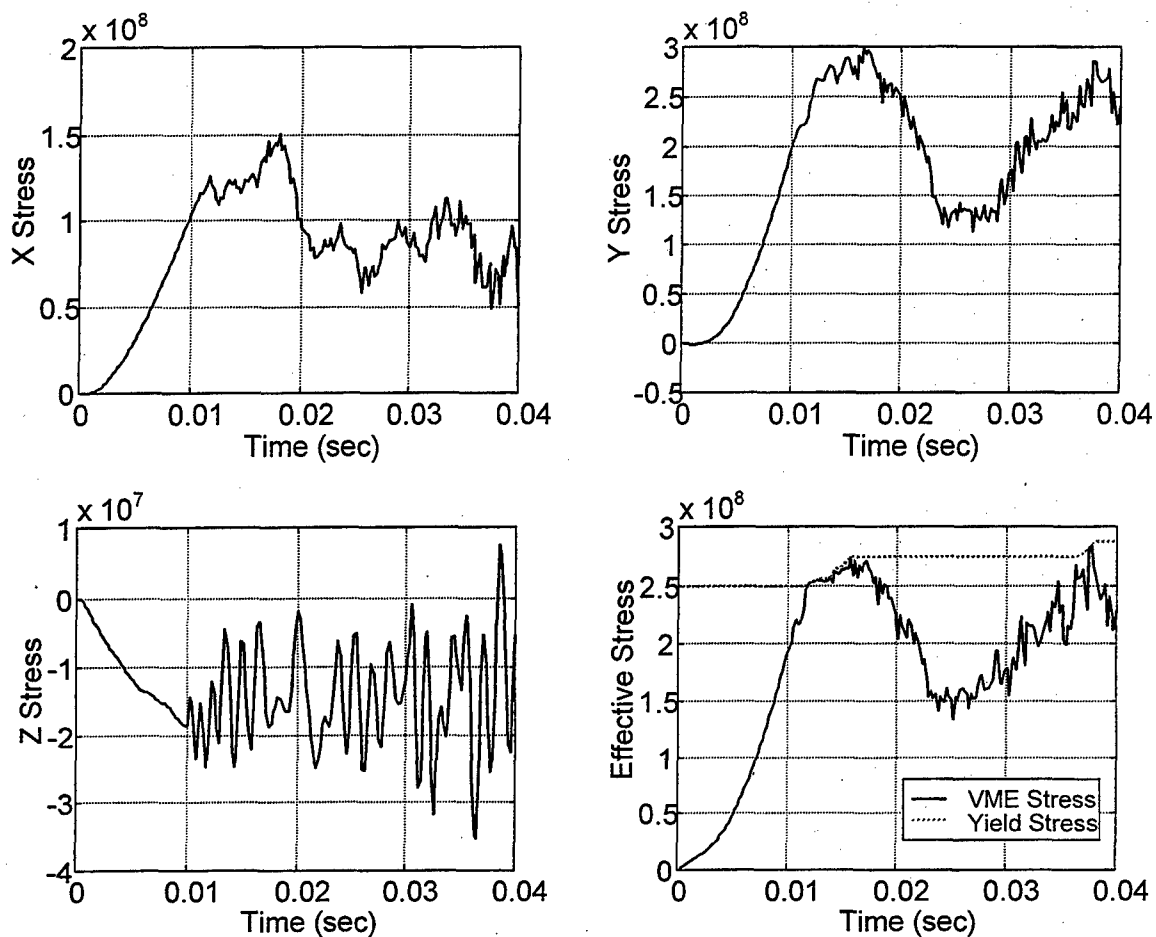


Figure 14. Stress Component Time History in Bottom Fiber: Pinched Cylinder, Void Effects and Drilling Moment Applied.

G. ELASTIC SPHERICAL CAP WITH A CENTER HOLE

The results of analysis with the shell element developed here are compared to results of the problem proposed in MacNeal and Harder [26]. The structure is a hemisphere of radius 10 units with a thickness of 0.04 units, and has an 18° hole cut in the center. Taking

advantage of the symmetry in the structure, only $\frac{1}{4}$ of the structure is modeled. An eight-by-eight mesh is used, for a total of 64 elements, with four integration points through the thickness of each element. The mesh used is shown in Fig. 15. The material properties of the structure are: $E = 6.825 \times 10^7$, $\rho = 0.001$, and $\nu = 0.3$. No void or drilling moment effects are employed in this example. Opposing forces of magnitude 2.0 are applied at each quadrant: the force at node 73 is of magnitude -1.0 and parallel to the y-axis, and the force at node one is of magnitude 1.0 and parallel to the x-axis. To obtain a representative static response using a dynamic model, the applied force begins with 0.0 magnitude at time 0 , and increases linearly with a rise time of 0.16 seconds to the specified value. A calculation time step of 1×10^{-6} seconds is used, with termination at 0.22 seconds. The maximum deflection of node one was 0.0929 , where the theoretical value is 0.0940 , for a normalized displacement of 0.988 . This is within the range of the results listed in MacNeal and Harder from the QUAD2 and QUAD4 elements, which are considered to be accurate for this problem [26]. These results are summarized in Table 9.

Table 9. Comparison of Results for Spherical Cap with Elastic Loading, and Results from MacNeal and Harder [26].

	New Shell Element	QUAD2	QUAD4
Normalized Displacement	0.988	0.986	1.005

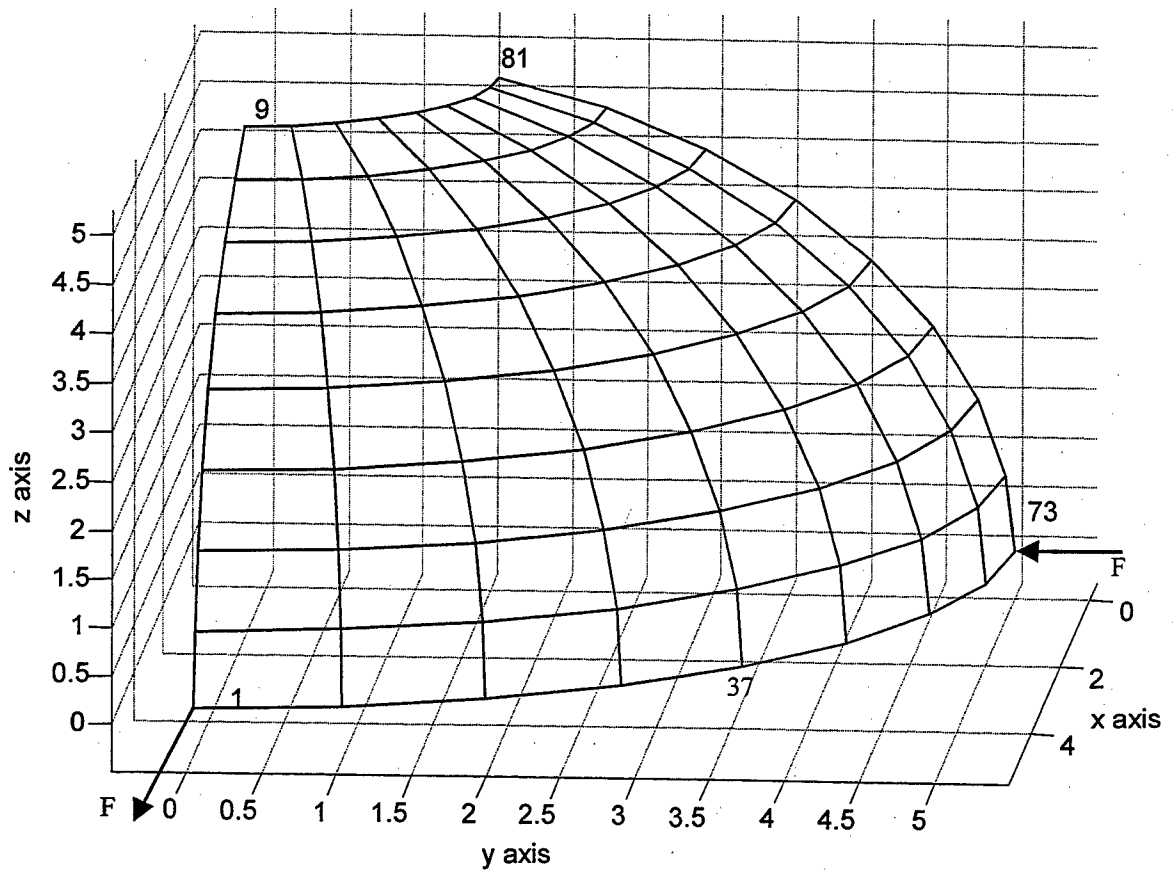


Figure 15. Mesh Structure for Spherical Cap.

H. SPHERICAL CAP WITH A CENTER HOLE, IMPACT LOADING

Key and Hoff [11] used the previous problem to test their element with an impact (step) load. The structure and material properties are the same, and the mesh is the same as shown in Fig. 15. The load is applied at its maximum at $t=0.0^+$ seconds, rather than ramped up. Figure 16, which plots the displacement of node one obtained from both the shell element presented here and the element developed by Key and Hoff [11], shows that the

results of this element are comparable to those obtained by other elements under impact loading.

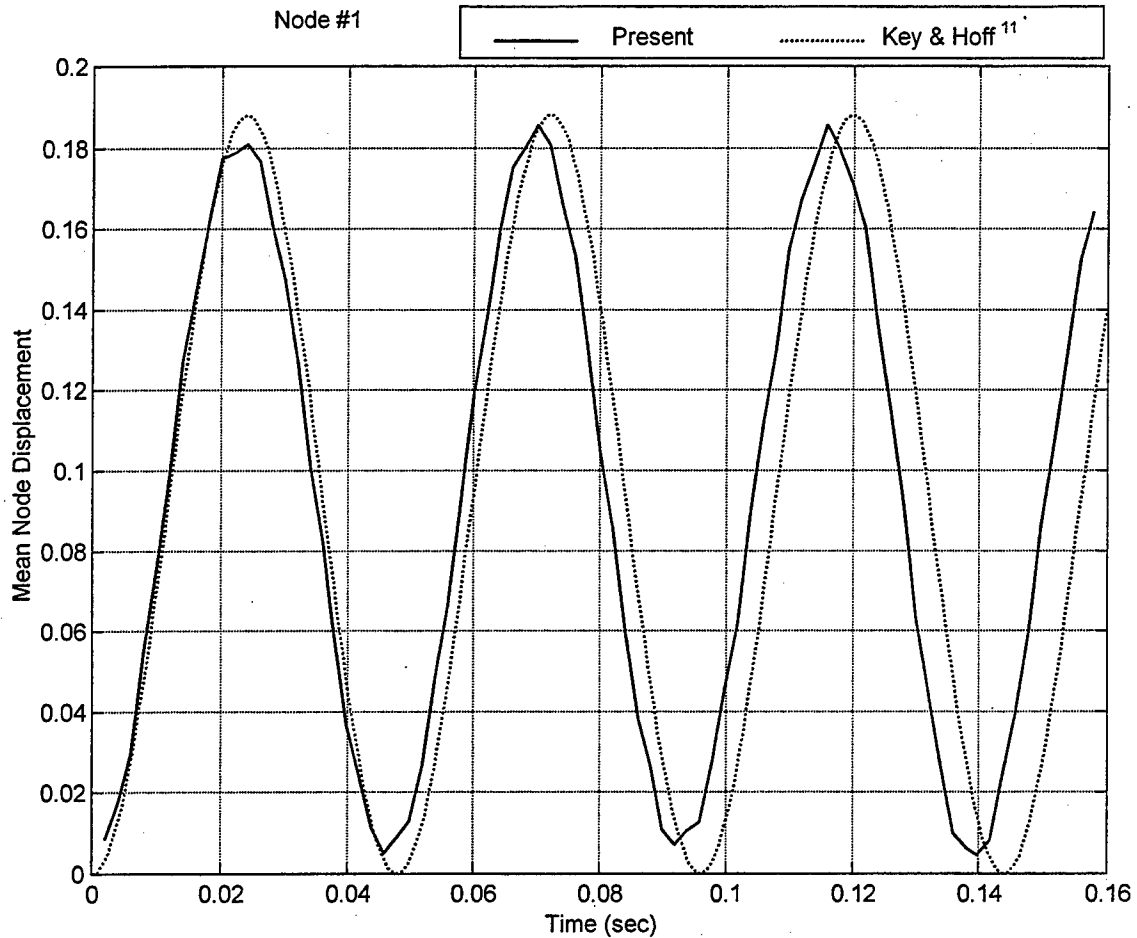


Figure 16. Node One Displacement Time History: Pinched Spherical Cap with Impact Loading.

I. SPHERICAL CAP WITH A CENTER HOLE, ELASTO-PLASTIC LOADING

The spherical cap structure shown in Fig. 15 is now used to verify the stability and convergence of the damage-constitutive equations under double curvature, and to illustrate

both the effects of void growth and nucleation and a drilling moment on a more complex structure. The material properties used are the same as shown in Tables 2 and 3. The structure has a radius of 5.0 meters and a thickness of 25 cm, for a radius to thickness ratio of 20:1, commonly considered the limit of thin-shell theory. Although a force is applied at each quadrant, all four loads are directed inwards. Therefore, only one load is applied at node 37 of the mesh shown in Fig. 15. The applied load is 5.9397×10^6 N, with a drilling moment of -7.4246×10^5 Nm applied for the cases indicated. A calculation time step of 1×10^{-5} seconds is used, with output every 5×10^{-4} seconds and stopping at 0.2 seconds. The load is initial zero, and increases linearly to its maximum at 0.01 seconds. The results shown in Table 10 are for the element nearest the applied load, where stress is maximum. The porosity versus effective plastic strain relationship is shown in Fig. 17. Due to the small amount of plastic strain, porosity is restricted to the linear zone, as shown. Figures 18a and 18b illustrate the effective stress versus effective strain in the same element on the compressive side and tensile side, respectively. Note that as the structure returns from a peak displacement, the stress follows the elastic modulus, not the tangent modulus. This reflects the correct behavior of material: the elastic modulus is not affected by plastic deformation, only the yield stress is affected. The node displacement where the force is applied is shown in Fig. 19. This figure shows that there are at least two vibration modes in operation.

Table 10. Summary of Results for Elasto-Plastic Spherical Cap.

Peak Values for Element #25	No Voids No DM	No Voids Drilling Moment	Voids No DM	Voids and Drilling Moment
σ_{VM} (MPa)	250.26	250.76	250.26	250.75
$\epsilon_{effective}$ ($\times 10^3$)	1.0730	1.1021	1.0731	1.1021
σ_{xx} (MPa)	202.09	199.44	202.08	199.43
σ_{yy} (MPa)	-47.093	-58.183	-47.087	-58.192
σ_{zz} (MPa)	-5.2570	-12.433	-5.2571	-12.433
$\epsilon^{plastic}$ ($\times 10^5$)	2.1002	4.1795	2.1020	4.1849
Φ (Porosity) ($\times 10^6$)	NA	NA	6.7149	12.763
Deflection (m) (Point of Loading)	0.0233	0.0246	0.0233	0.0246

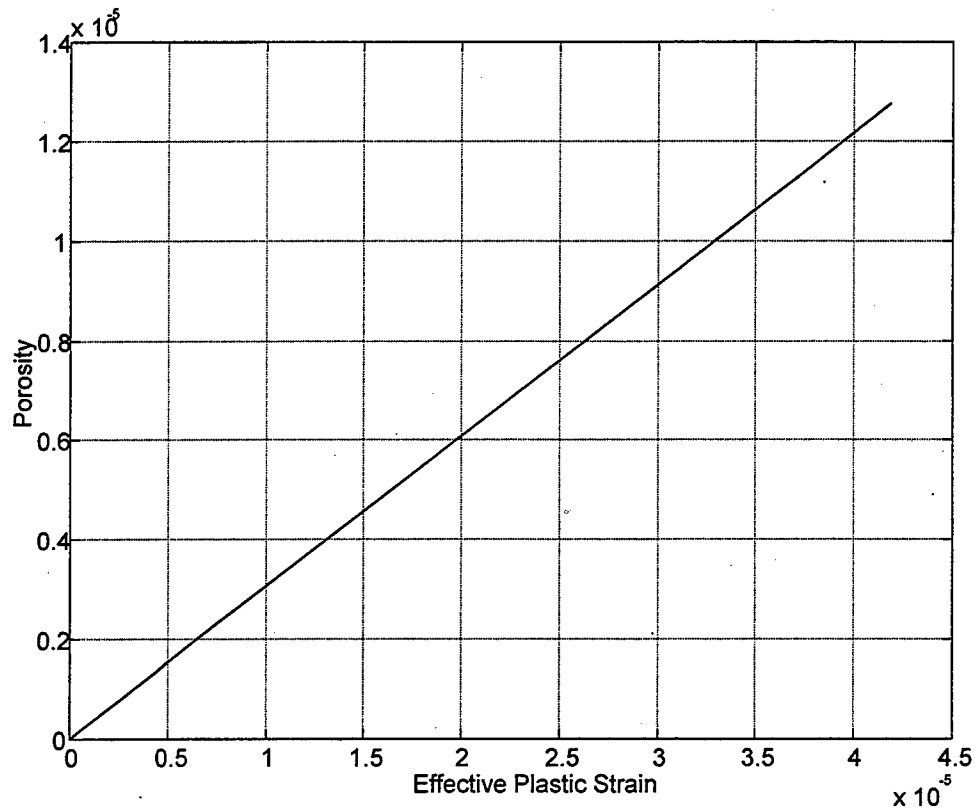


Figure 17. Porosity versus Effective Plastic Strain in Inner Fiber: Spherical Cap with Void and Drilling Moment Effects.

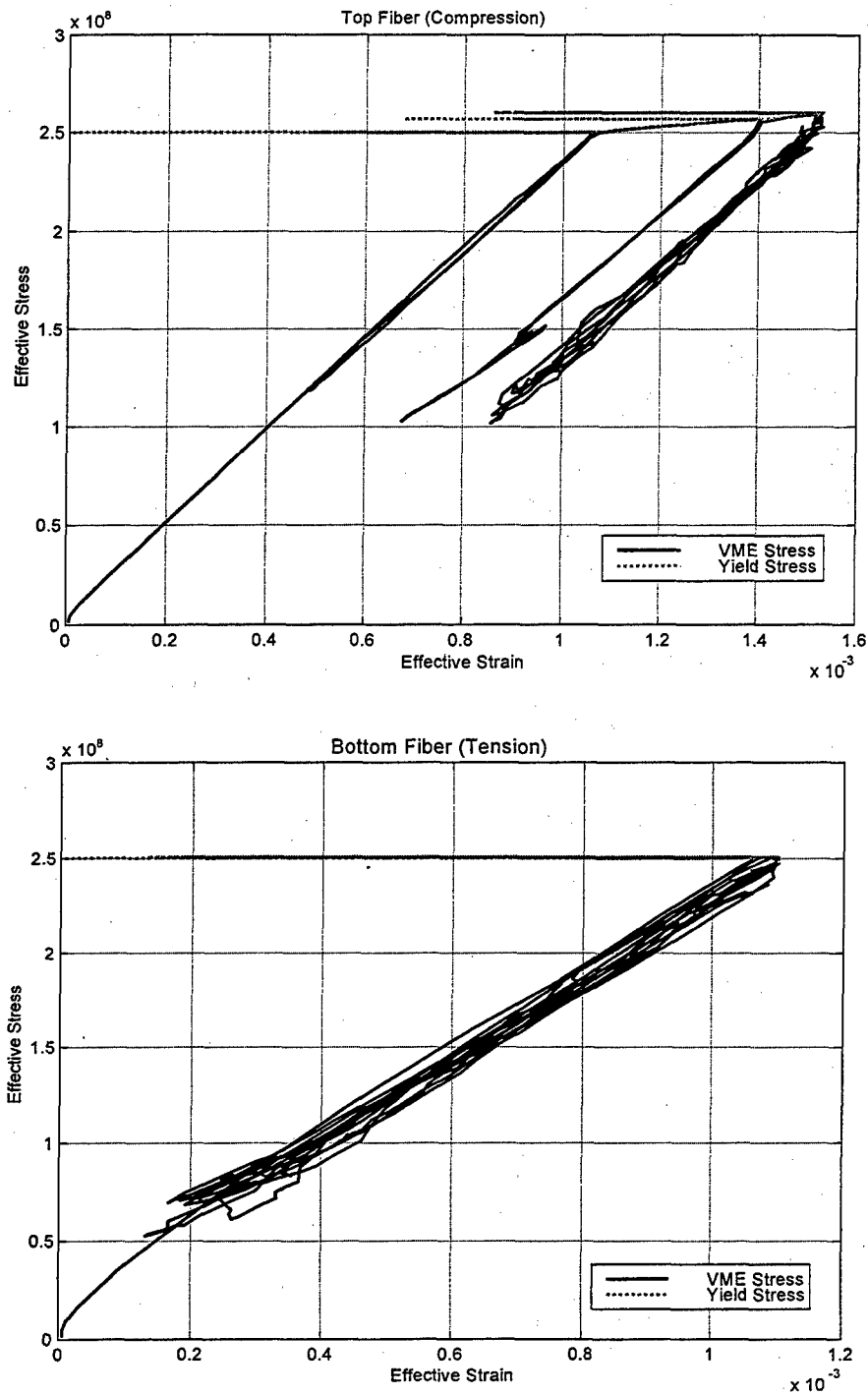


Figure 18. Comparison of Void and Drilling Moment Effects in a) Compression (Top) and b) Tension (Bottom) for Spherical Cap.

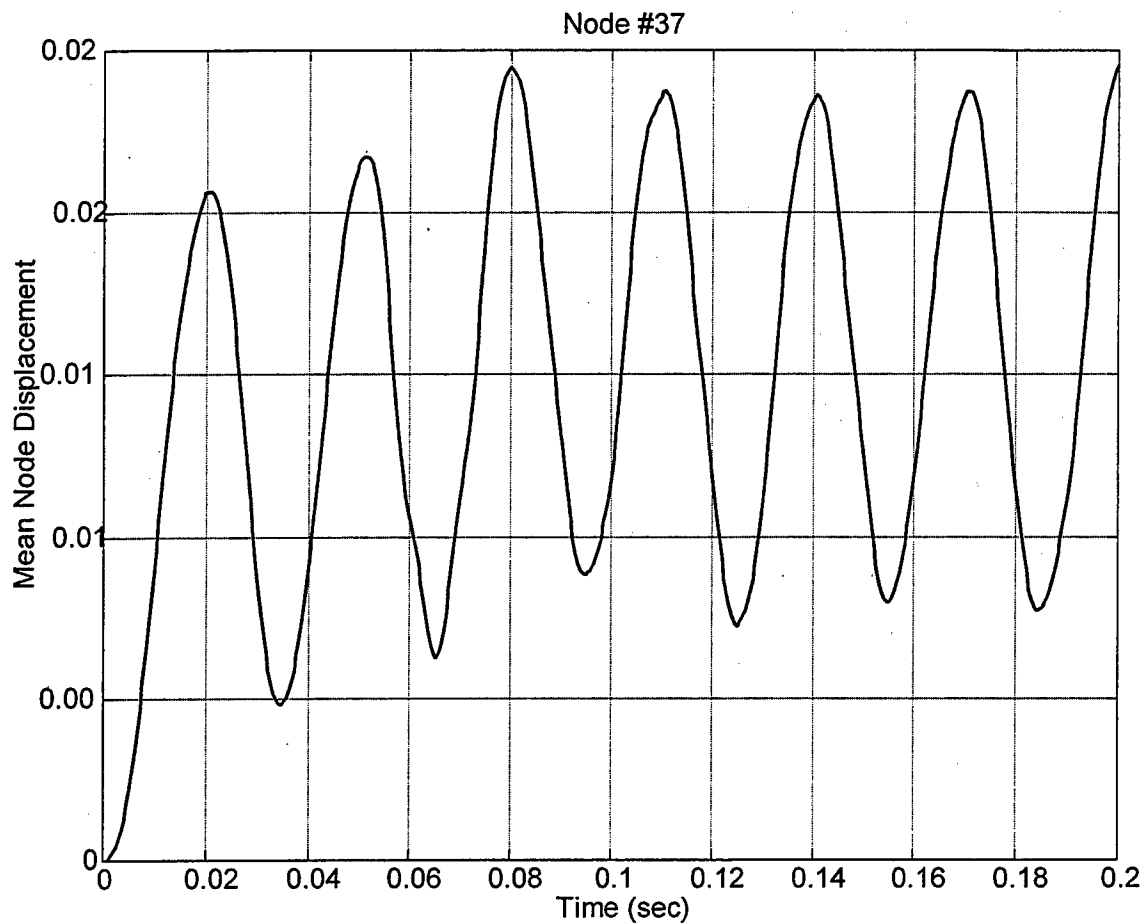


Figure 19. Contact Node Mean Displacement Time History: Spherical Cap with Void and Drilling Moment Effects.

During testing, it became apparent that this problem is not suitable for testing elasto-plastic analysis. Since the structure is concave, the entire structure quickly collapses once yielding begins. In addition, the single point used to prevent rigid-body motion also provided a stress concentration and additional yielding (the "corner" began folding over).

This necessitated using a force that just causes plastic flow, but does not collapse the

structure or cause yielding near the anchored node. This is illustrated by an analysis of a 5 percent greater load than used for the above analysis. The resulting node 37 displacement is shown in Fig. 20, and the deformed structure in Fig. 21. However, it is still apparent that the qualitative results obtained in the previous elasto-plastic examples carried through to this problem; both voids and the drilling moment decreased stress in fibers under tension, and increased stress in fibers under compression. Due to the small amount of plastic strain, the increase in porosity was extremely small, which minimized the effects of the voids.

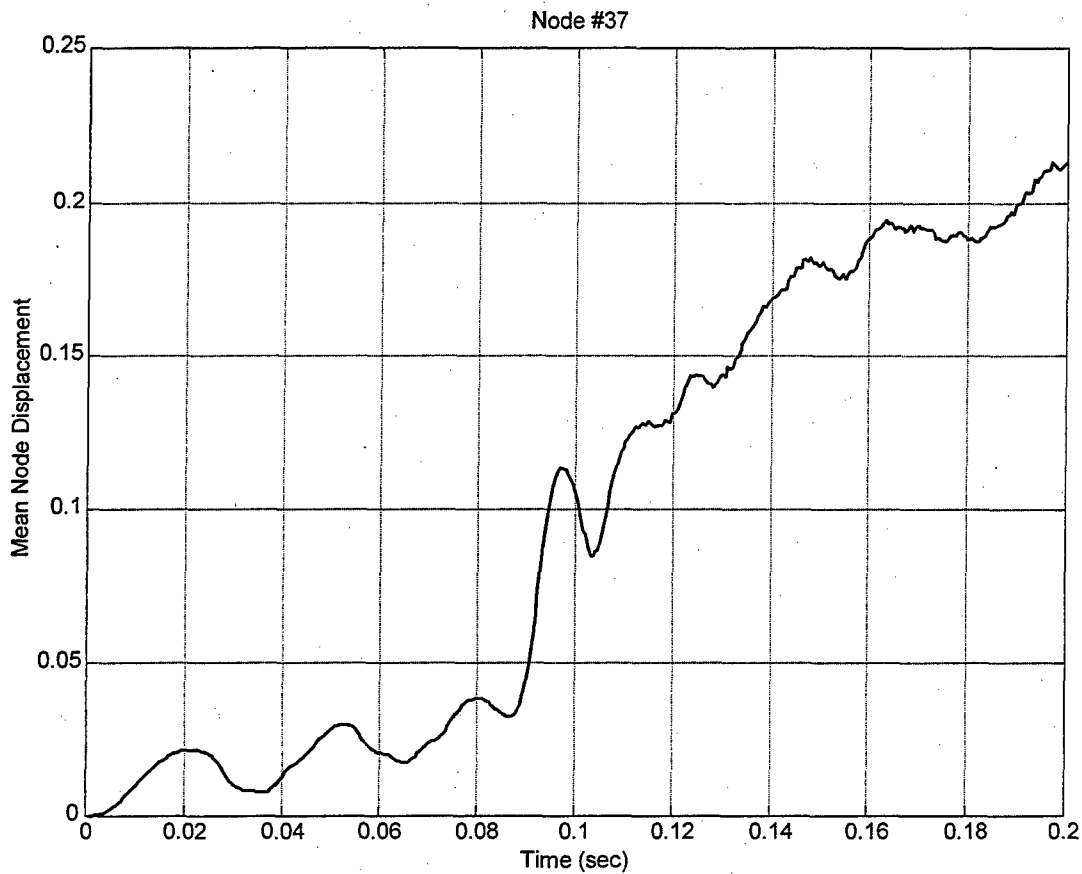


Figure 20. Contact Node Displacement: Spherical Cap with 5% Greater Load.

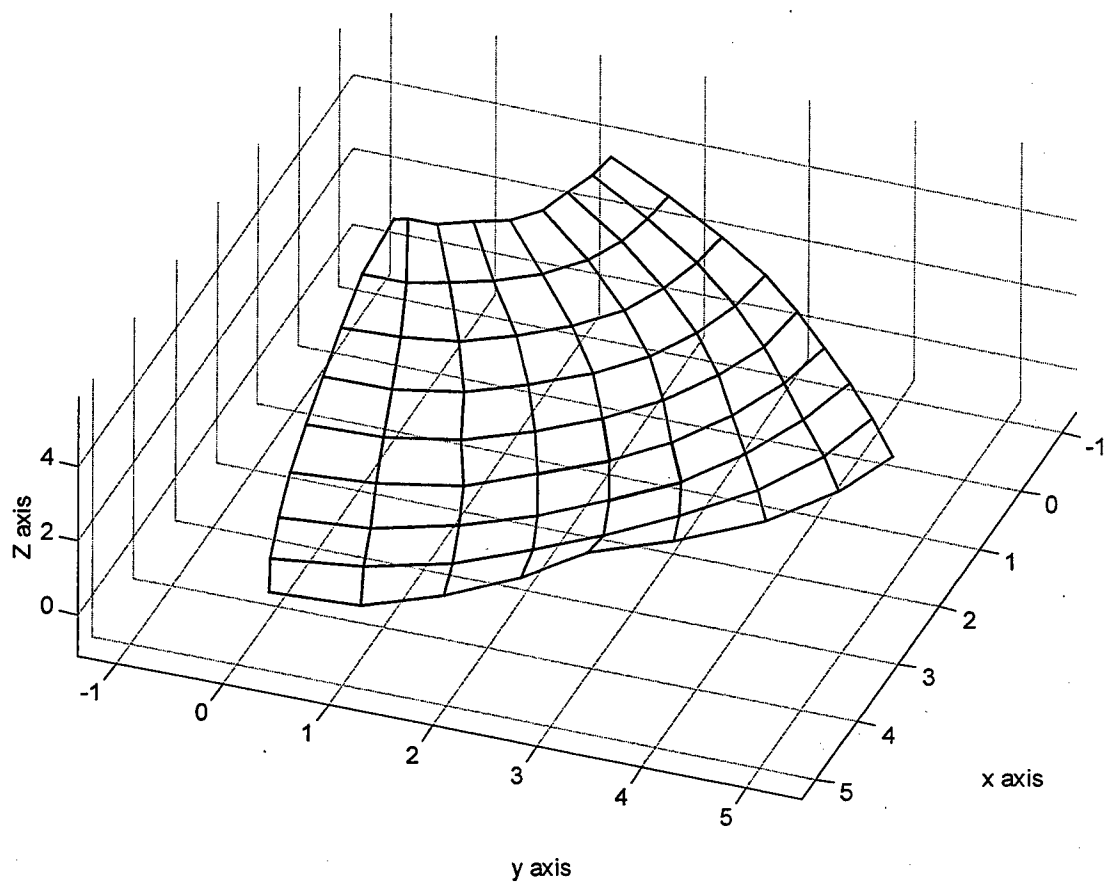


Figure 21. Deformed Structure at End of Analysis: Spherical Cap with 5% Greater Load.

V. DYSMAS IMPLEMENTATION

After verifying the formulation using in-house code (see previous section and McDermott and Kwon [27]), the next step was to implement the formulation into DYSMAS, using source code provided by NSWCCardero. This section will discuss the details of implementing this shell formulation into the DYSMAS source code.

Since DYSMAS treats the constitutive model, or material model, separate from the element formulation, it was necessary to conduct the implementation in two steps: 1) implement the new shell formulation and verify, 2) implement Gurson's void model as a new material type and verify. Since the source code provided was not fully functional, several corrections were required before completing the shell formulation implementation. These corrections are summarized in Appendix A.

A. GENERAL IMPLEMENTATION ISSUES

The in-house code used to test the formulation was linear, and the integration scheme was to step through all the elements of the same type, with an inner loop for the integration points. DYSMAS is non-linear (based on strain increments), and divides each element type into groups of 32, or some other constant that is determined before compiling (variable NLQ). In addition, DYSMAS steps through each integration point, with an inner loop for each element in the group. This required substantive modification to the in-house implementation. The goal of this implementation is to fully support all functions of DYSMAS. The areas where this implementation falls short will be detailed later.

B. SHELL FORMULATION

The formulation is implemented by providing for the following functions: data input and echo, initialization, computation, and data output. All new subroutines are listed in Appendix B. Changes made to original subroutines are highlighted in Appendix C. The following subroutines required modification in order to support the formulation:

DYNAI - read new formulation number (8), and write to echo file

ELEM2D - calls the new formulation during the solution phase

The following new subroutines have been implemented:

KWNMCD - main algorithm for new shell formulation

KMTRAN - computes variables for traditional hourglass control and element area

KMCON - call appropriate material model, including Gurson's Void Model

KMFRC - computes hourglass force and puts internal forces into system matrix

KMDRILL - computes drilling moment for applied forces, including contact

KMCPDP - copies displacements to/from global variables to/from local variables

KMSHAP - 1-D shape function

KMINV3 - explicit 3 by 3 matrix inversion

All subroutines have been commented to explain the details of the algorithm, so they will not be repeated here.

C. MATERIAL MODEL

The implementation of Gurson's void model required modification to the following subroutines:

BLKDAT - number of material constants for new model

IN3DIS - initialization

MATIN - read new material model from input data file

NBSINT - initialization

PENSTF - initialization

STIFFS - initialization

STIFFSN - initialization

PRINTM - echo of input material properties

SCA_ASC - output of data in ASCII format

SCA_DYS - output of data in DYSMAS format

SCA_TEC - output of data in TECPLOT format

The following new subroutines were added for the new material model:

SETS44 - initialize material variables and AUX14 variables

SCA_GET - provide additional contour plot information

SHL44S - new material model

Note that SCA_GET will also support providing contour plot information for any variable in AUX14 of other formulations, but this has not been implemented. Draft pages

for addition to the user's guide covering the new material model are provided in Appendix D. Gurson's void model is implemented as material type number 44. The current implementation does not allow this material type to be used with any other element type.

D. IMPLEMENTATION ISSUES

The following issues have not been resolved for the new implementation:

1. The indexing for updating shell thickness is incorrect, and causes a memory violation. The option of updating the shell thickness for the new formulation should not be used until this is corrected.
2. The calculation of the drilling moment utilizes some FORTRAN 90 commands, and needs to be revised. A better implementation would place these calculations in FEM3D, prior to calling the appropriate formulation.
3. There is no theoretical basis for the critical time step calculation of this element, even though the method used has survived extensive testing. This involves modifying the computed surface area with a normalized thickness and a constant (see KMTRAN in Appendix B). It is certain that since this element provides full integration through the thickness, the thickness affects the critical time step, but not in a linear manner. Basing the time step on the thickness alone is far too conservative, while neglecting the thickness frequently causes an unstable solution. The method shown in KMTRAN's listing was used with success for all verification problems.
4. The method of hourglass control is time-consuming and inefficient. However, this

method works when the other methods available in DYSMAS do not. This method also works for other formulations, but is currently available only in the new formulation. The current implementation in KMFRC disregards the choice of hourglass control specified in the input file, strictly using the new method. This needs to be revised, and using a more efficient formulation for hourglass control would greatly improve efficiency.

5. Failure is not implemented in either the shell formulation, or in the material model.

This is critical, and should be the next step in implementation.

6. The subroutines listed in the appendices are "first drafts." Although functional, no emphasis was placed on efficiency and/or speed. There is certainly room for improvement in this area.

7. Dynamic Relaxation does not work with the new material model. This is probably caused by faulty energy calculation in SHL44S, and must be corrected.

The current implementation is functional, and several verification problems were analyzed.

VI. DYSMAS VERIFICATION

The following problems were analyzed using the DYSMAS implementation described in the previous section. Since the DYSMAS preprocessor and postprocessor are not available, both a preprocessor and postprocessor that use the ASCII format were created using MATLAB. All associated script files are available upon request. In addition, TECPLOT was used to show contour plots and animate structural response.

A. ELASTIC CANTILEVER PLATE WITH SMALL DISPLACEMENT

A cantilever plate with small displacement that ensured all stresses were below the yield stress was tested using both the in-house code (FEA) and DYSMAS with the Belytschko-Tsay, Hughes-Liu, QPHM, and the formulation presented herein (referred to as the Kwon-McDermott element). The Kwon-McDermott element used the new material model; Gurson's Void model. An analytic solution is available for this problem, and the results are shown in Table 11. Although the Kwon-McDermott element obtains an answer closest to the analytic solution, this will not always be the case. The purpose of this example is only to show that the algorithm has been correctly implemented for the purely elastic case.

Table 11. DYSMAS Solution of Cantilever Plate.

Solution Method	Displacement ($\times 10^{-6}$ m)	Error from Analytic (%)
Analytic	58.6	NA
FEA (in-house)	55.368	11.54
Belytschko-Tsay	56.788	3.09
Hughes-Liu	56.788	3.09
QPHM	57.271	2.27
Kwon-McDermott	57.661	1.60

This problem was also used to verify that the element works correctly across more than one group. A simple four element mesh was used to obtain the results shown in Table 11. The same problem was solved using a 100 element mesh, and the Kwon-McDermott element obtained the same answer; 57.661×10^{-6} m. This is reasonable since the deflection is so small. This problem was also used to verify the solution using the top and bottom surfaces, vice the mid-plane, as the reference surface. The element obtained the same answer for both the top and bottom reference surfaces: 57.662×10^{-6} m. As mentioned in the previous section, the tests on dynamic relaxation and updating the element thickness both failed. Tests on the basis for the sound speed passed for all three options available.

The tests using this example were designed to identify improper coding, and the results show that the element is functional, but the calculations relating to dynamic relaxation and thickness update contain some errors. The same geometry was used for a different set of dimensions and applied force using the Belytschko-Tsay and the Kwon-McDermott Element. The results are shown in Figure 22.

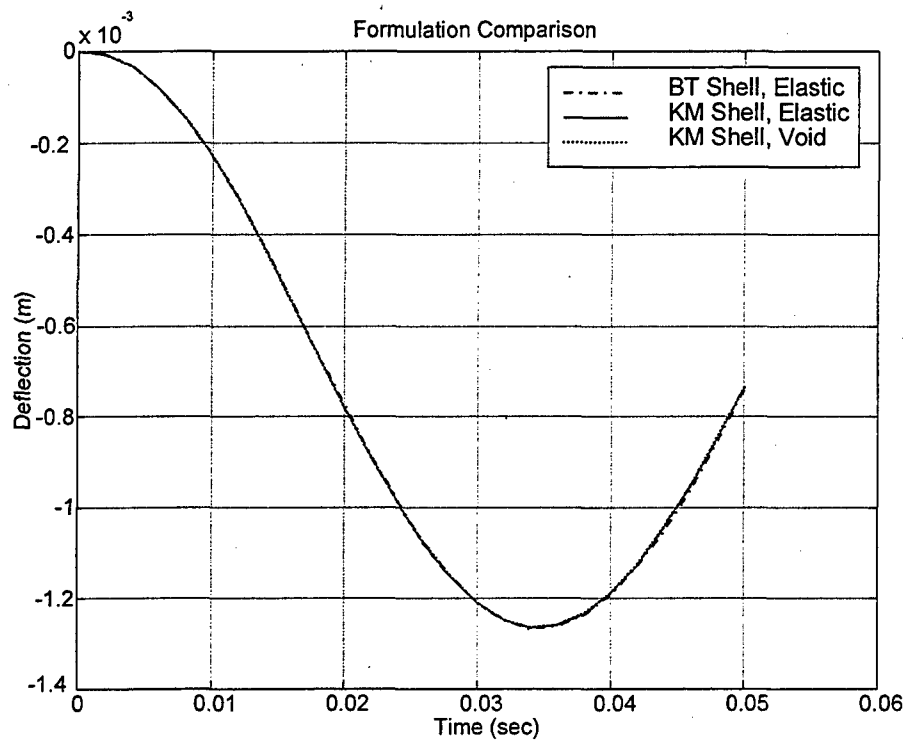


Figure 22. Comparison of Element Formulations for Cantelever Plate.

B. SINGLE CURVATURE VERIFICATION: ELASTIC-PLASTIC CYLINDER

The same elastic pinched cylinder used to verify the in-house code was used with the Kwon-McDermott element in DYSMAS (See Table 7). Figure 23 shows the first run, using a 1/4 cylinder model with appropriate symmetry boundary conditions. This figure also verifies the changes made to the TECPLOT output subroutines, along with other contour plots and animations (animations available upon request).

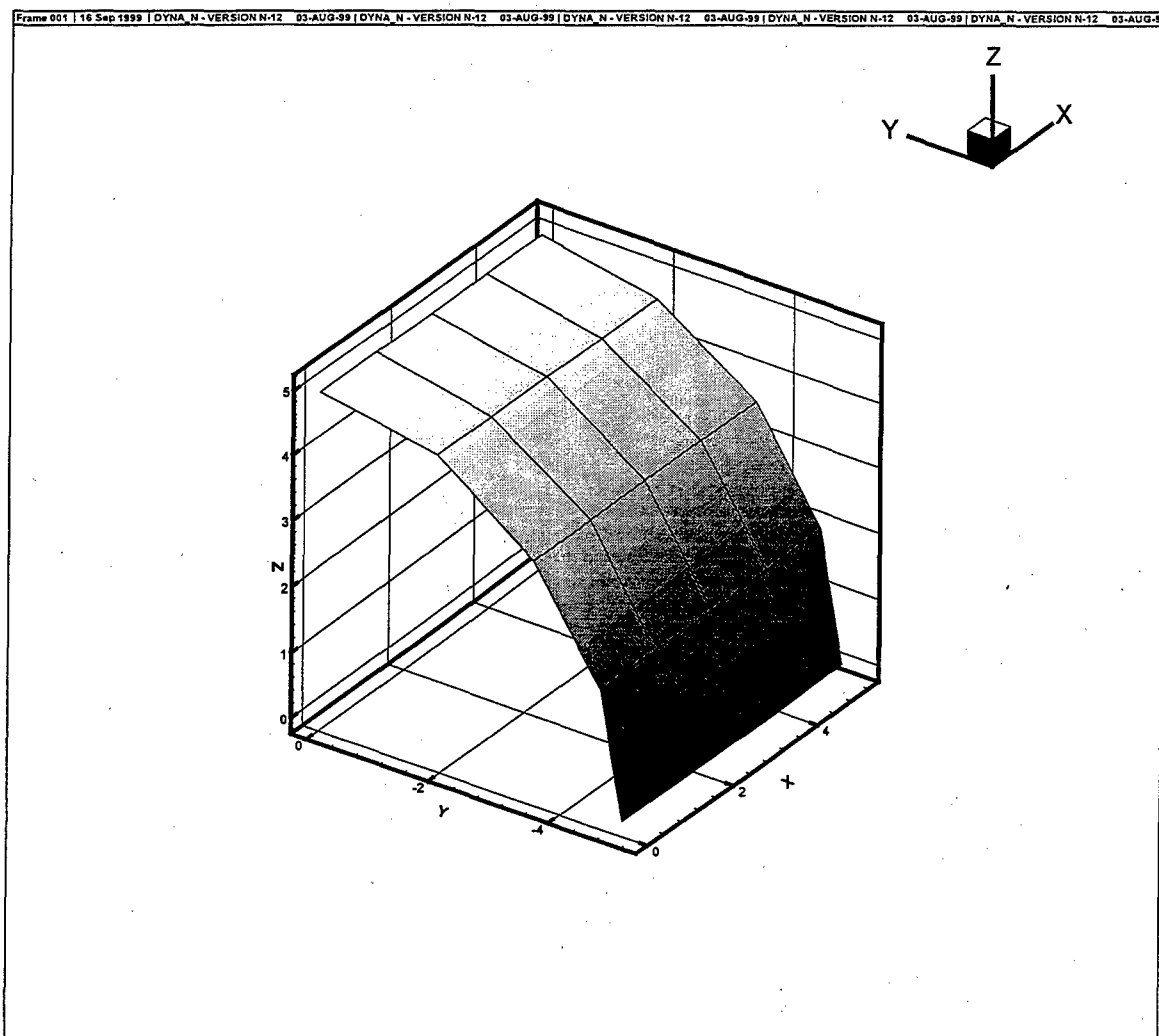


Figure 23. Mesh Structure for Pinched Cylinder using TECPLOT.

Table 12 compares the results from the mesh shown in Fig. 23 with the results shown in Table 7. The improvement is due to the non-linear solution method used by DYSMAS, vice the linear solution method used in the in-house code.

Table 12. DYSMAS Elastic Pinched Cylinder Results.

Solution Method	Radial Contraction (m)	Error from Analytic (%)
Analytic (Twice Static)	0.2234	NA
FEA (16 Element Mesh)	0.1995	10.7
Kwon-McDermott (16 Element Mesh, DYSMAS)	0.2135	4.4

Figure 24 shows an elastic-plastic pinched cylinder at the end of a solution run. A one-half structural mesh is used to magnify any problems in the hourglass mode control method. All other hourglass control methods available in DYSMAS failed this test, but as Fig. 24 shows, the new method is effective. Since there is no comparison data for this problem, the displacement and stress field will not be discussed.

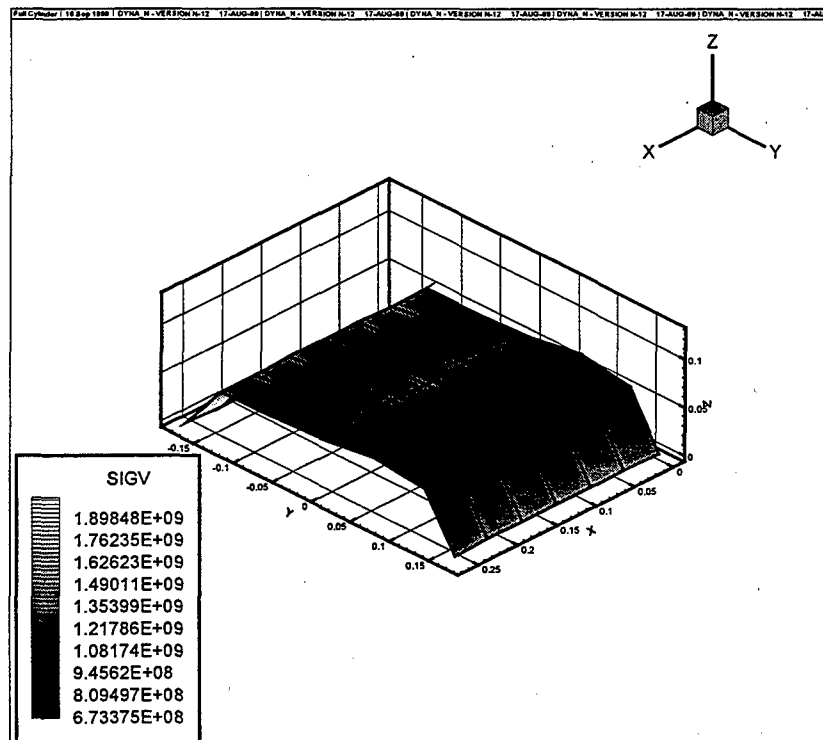


Figure 24. Elastic-Plastic Pinched Cylinder using DYSMAS.

C. BALL IMPACT PROBLEM

A test problem received from NSWCC Carderock involved a solid ball striking a fully clamped plate. The original plate was too thick for adequate modeling using shell elements, as shown in Fig. 25. The characteristic length to thickness ratio for this problem was 5:1.

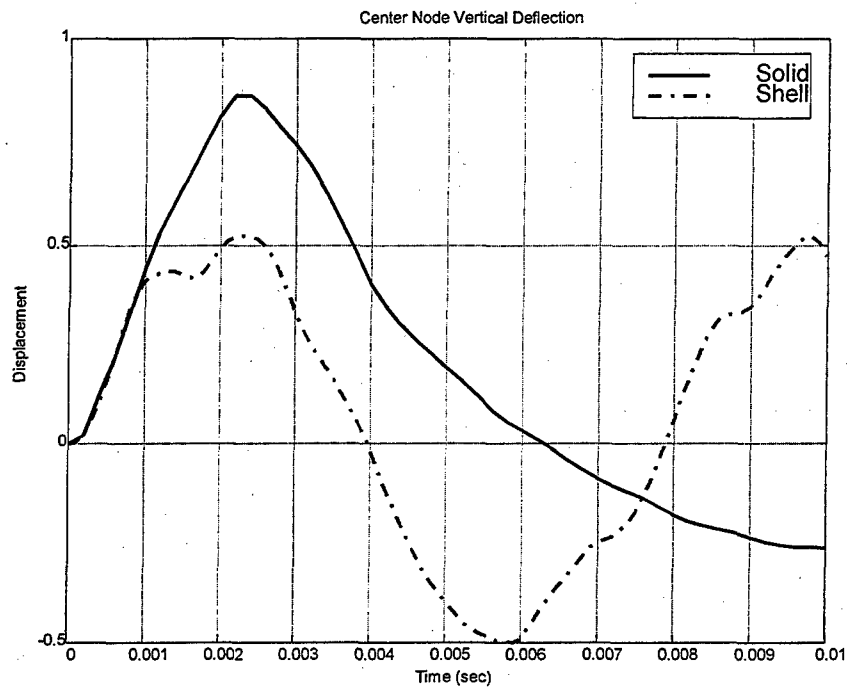


Figure 25. Ball Impact Problem Center Node Displacement with 5:1 Aspect Ratio.

The problem structure modeled with all solid elements at the end of the solution is shown in Fig. 26. The same problem with the plate modeled using the new shell element is shown in Fig. 27. The center node displacement time history is shown in Fig. 28, which also include the solution for the Belytschko-Tsay element. The results for this problem should be the same for both shell formulations. The additional rotational degrees of freedom in shell

69

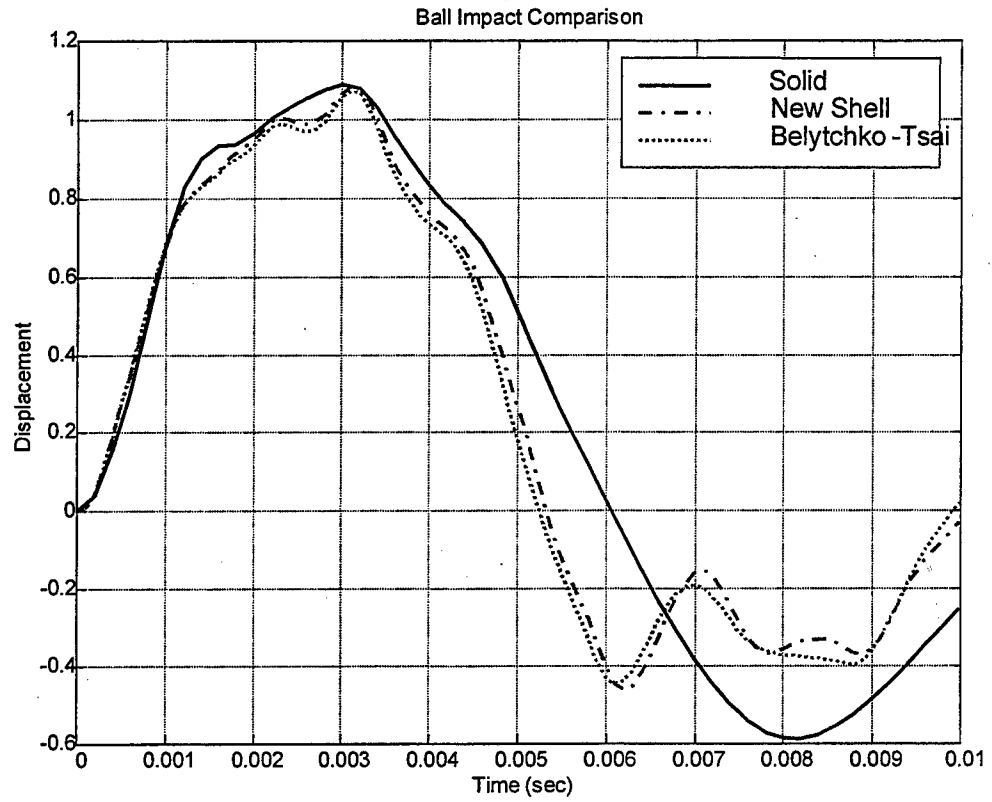


Figure 28. Center Node Displacement Comparison for Ball Impact Problem.

For the purely elastic case, both the element formulation and the material model provide correct answers for all test problems. Testing in the elastic-plastic region is incomplete.

VII. CONCLUSIONS AND RECOMMENDATIONS

A new shell formulation was developed for transient dynamic analysis which includes both void effects with plastic deformation and the transverse normal stress with drilling moment. The element is compatible with most shell elements which have three translation and three rotation degrees of freedom per node.

A numerically stable scheme was developed for Gurson's nonlinear constitutive equation model. The model includes both void nucleation and void growth. Furthermore, hourglass control was implemented into the algorithm to avoid spurious modes caused by the under-integration scheme.

The drilling moment was important for thick plates and shells. It induced large transverse normal stress and affected the plastic deformation. Similarly, the effect of voids on plastic deformation was not negligible. Numerical studies show that the effective plastic strain increased up to fifteen percent due to the combined effects of voids and the transverse normal stress.

The element presented here passed all testing when implemented into an in-house finite element analysis code. The element also passed all testing in the elastic region when implemented into DYSMAS. Testing in the elastic-plastic region was not complete at the time of this report. The majority of code modification and new coding required to implement the new element into DYSMAS is complete, although there are a few issues that must be resolved. The current DYSMAS implementation of this element does not support updating element thickness, dynamic relaxation, or element failure. In addition, there are several areas

where improvements in efficiency can be made: drilling moment calculation, critical time step calculation, hourglass control, and main algorithm calculation. Material data indicating the proper values for the constants in Gurson's void model are required, as are structural test results in the elastic-plastic region.

APPENDIX A. CORRECTIONS IMPLEMENTED INTO DYSMAS.

FEM3D	Corrected print interval calculation which caused all output files to be written at each calculation time step.
PRTDAT	Corrected multiple file writes at each print step.
SCA_TEC	Corrected incorrect use of a scratch file.
TEC_TEN	Corrected incorrect use of a scratch file.
TECPLOT	Assigned text to current zone to prevent all text being written to base frame.

These corrections are included in Appendix C, and comments in the source code detail the changes made.

APPENDIX B. NEW DYSMAS SUBROUTINE LISTINGS.

(The following three subroutines are in the file "kmaux.for")
 c Following Subroutines used for Kwon-McDermott Shell Element
 c kminv3 - computes inverse of a 3 by 3 matrix (explicitly)
 c kmshap - returns 1-D shape function values at zeta
 c kmcpdp - copies displacements to/from global variables
 c McDermott 1999

```

subroutine kminv3(a, ainv,det)
c Direct calculation of 3x3 matrix inverse
implicit double precision (a-h,o-z)
dimension a(3,3),ainv(3,3)

ainv(1,1) = a(2,2)*a(3,3) - a(3,2)*a(2,3)
ainv(2,1) = -a(2,1)*a(3,3) + a(3,1)*a(2,3)
ainv(3,1) = a(2,1)*a(3,2) - a(3,1)*a(2,2)
ainv(1,2) = -a(1,2)*a(3,3) + a(3,2)*a(1,3)
ainv(2,2) = a(1,1)*a(3,3) - a(3,1)*a(1,3)
ainv(3,2) = -a(1,1)*a(3,2) + a(3,1)*a(1,2)
ainv(1,3) = a(1,2)*a(2,3) - a(2,2)*a(1,3)
ainv(2,3) = -a(1,1)*a(2,3) + a(2,1)*a(1,3)
ainv(3,3) = a(1,1)*a(2,2) - a(2,1)*a(1,2)

det = a(1,1)*ainv(1,1) + a(1,2)*ainv(2,1)
1 + a(1,3)*ainv(3,1)

do 20 j = 1,3
do 10 i = 1,3
10 ainv(i,j) = ainv(i,j)/det
20 continue
return
end

subroutine kmshap(r,shapef)
c returns 1D shape function
implicit double precision (a-h,o-z)
dimension shapef(2)

shapef(1) = 0.5 * (1 - r)
shapef(2) = 0.5 * (1 + r)
return
end

subroutine kmcpdp(displ,idir,ie)
c Copy displacements to a local vector
c idir = 1, copy to edisp, idir=0, copy from edisp
implicit double precision (a-h,o-z)
include 'nlqpar.inc'
common/bk02/iburn,ismo,dt1,dt2
common/aux10/area(nlq),
1 px1(nlq),px2(nlq),px3(nlq),px4(nlq),
& px5(nlq),px6(nlq),px7(nlq),px8(nlq),
2 py1(nlq),py2(nlq),py3(nlq),py4(nlq),
& py5(nlq),py6(nlq),py7(nlq),py8(nlq),
3 pz1(nlq),pz2(nlq),pz3(nlq),pz4(nlq),
& pz5(nlq),pz6(nlq),pz7(nlq),pz8(nlq),

```

```

4 dx1(nlq),dx2(nlq),dx3(nlq),dx4(nlq),
5 dx5(nlq),dx6(nlq),dx7(nlq),dx8(nlq),
6 dy1(nlq),dy2(nlq),dy3(nlq),dy4(nlq),
7 dy5(nlq),dy6(nlq),dy7(nlq),dy8(nlq),
8 dz1(nlq),dz2(nlq),dz3(nlq),dz4(nlq),
9 dz5(nlq),dz6(nlq),dz7(nlq),dz8(nlq)
common/aux12/
1 wxx1(nlq),wxx2(nlq),wxx3(nlq),wxx4(nlq),
2 wyy1(nlq),wyy2(nlq),wyy3(nlq),wyy4(nlq),
3 wzz1(nlq),wzz2(nlq),wzz3(nlq),wzz4(nlq),
4 a13(nlq),a23(nlq),a33(nlq)
dimension disp(*)

```

```

if (idir.eq.1) then
  disp(1) = dx1(ie)
  disp(2) = dy1(ie)
  disp(3) = dz1(ie)
  disp(4) = wxx1(ie)
  disp(5) = wyy1(ie)
  disp(6) = wzz1(ie)
  disp(7) = dx2(ie)
  disp(8) = dy2(ie)
  disp(9) = dz2(ie)
  disp(10) = wxx2(ie)
  disp(11) = wyy2(ie)
  disp(12) = wzz2(ie)
  disp(13) = dx3(ie)
  disp(14) = dy3(ie)
  disp(15) = dz3(ie)
  disp(16) = wxx3(ie)
  disp(17) = wyy3(ie)
  disp(18) = wzz3(ie)
  disp(19) = dx4(ie)
  disp(20) = dy4(ie)
  disp(21) = dz4(ie)
  disp(22) = wxx4(ie)
  disp(23) = wyy4(ie)
  disp(24) = wzz4(ie)
else
  dx1(ie) = disp(1)
  dy1(ie) = disp(2)
  dz1(ie) = disp(3)
  wxx1(ie) = disp(4)
  wyy1(ie) = disp(5)
  wzz1(ie) = disp(6)
  dx2(ie) = disp(7)
  dy2(ie) = disp(8)
  dz2(ie) = disp(9)
  wxx2(ie) = disp(10)
  wyy2(ie) = disp(11)
  wzz2(ie) = disp(12)
  dx3(ie) = disp(13)
  dy3(ie) = disp(14)
  dz3(ie) = disp(15)
  wxx3(ie) = disp(16)
  wyy3(ie) = disp(17)
  wzz3(ie) = disp(18)
  dx4(ie) = disp(19)
  dy4(ie) = disp(20)
  dz4(ie) = disp(21)

```



```
wxx4(ie) = disp(22)
wyy4(ie) = disp(23)
wzz4(ie) = disp(24)
endif
return
end
```



```

    call shl4s (cm,a(ntmp0+1),a(n19))
elseif (mte.eq.12) then
    call shl12s (cm,capa)
elseif (mte.eq.15) then
    call shl15s (cm,capa)
elseif (mte.eq.18) then
    call fem18s (cm,capa)
elseif (mte.eq.19) then
    if (miter.eq.0) then
        call shl9sc (cm,a(n8),a(n9),capa)
    elseif (miter.eq.1) then
        call shl19s (cm,a(n8),a(n9),capa)
    elseif (miter.eq.2) then
        call shl9si (cm,a(n8),a(n9),capa)
    endif
elseif (mte.eq.21) then
    call shl21s (cm,capa,a(ntmp0+1),a(n19))
elseif (mte.eq.22) then
    call shl22s (cm,capa)
elseif (mte.eq.23) then
    lthrpr=nint(cm(48,mxe))
    call shl23s (cm,capa,a(ntmp0+1),a(n19),a(lthrpr),
1      a(n8),a(n9),a(lc11))
elseif (mte.eq.24) then
    if (miter.eq.0) then
        call sh24sc (cm,a(n8),a(n9),capa)
    elseif (miter.eq.1) then
        call shl24s (cm,a(n8),a(n9),capa)
    elseif (miter.eq.2) then
        call sh24si (cm,a(n8),a(n9),capa)
    endif
elseif (mte.eq.28) then
    call shl28s (cm,capa)
elseif (mte.eq.30) then
    call shl30s (cm,capa)
elseif (mte.eq.33) then
    call shl33s (cm,capa)
elseif (mte.eq.34) then
    call shl34s (cm,capa)
elseif (mte.eq.35) then
    if (miter.eq.0) then
        call shl35s (cm,a(n8),a(n9),capa)
    elseif (miter.eq.1) then
        call shl35s (cm,a(n8),a(n9),capa)
    elseif (miter.eq.2) then
        call shl35s (cm,a(n8),a(n9),capa)
    endif
elseif (mte.eq.38) then
    call shl38s (cm,capa)
elseif (mte.eq.39) then
    call shl39s (cm,capa)
elseif (mte.eq.41) then
    call shl41s (cm,capa)
elseif (mte.eq.42) then
    call shl42s (cm,capa)
elseif (mte.eq.44) then
    call shl44s (cm,capa)
else
    write(39,1000)
    write(13,1000)

```

```

      call adios(2)
1000 format(/5x,'*** illegal material for kwon-mcdermott shell ***',
1      /5x,'      execution aborted ')
      endif

c Calculate elastic sig3 for types that do not support KM element
      if (mte.ne.44) then
        do 20 i=lft,llt
          d3(i) = d3t(i)
          sig3(i) = sg3(i) + ym*d3(i)
20      continue
        endif

c Store new stress tensor for current ipt
      call tbsc2s (nmtcon,auxvec(lavloc),lav,nip*nmtcon,nip,ipt)

c Apply Rayleigh Damping
      if (dampk.ne.0.) call rydmp2(dampk,ym,pr)
      return
end

```

```

      subroutine kmdrill(e,f,ndlist,nlstm)
c *****
c Calculates Drilling Moment associated with external forces
c for Kwon-McDermott shell element
c (Does this by calculating a normal vector at each node,
c then finding the dot product with the applied forces)
c McDermott 1999
c Variable Listing
c e - nodal force array
c f - nodal moment array
c ndlist - array of nodes affected by this element group
c nlstm - number of entries in ndlist
c *****
      implicit double precision (a-h,o-z)

      include 'nlqpar.inc'

      common/bk00/numnp,numpc,numlp,neq,ndof,nlcur,numcl,numvc,
1 ndtpts,nelmd,nmmat,numelh,numelb,numels,numelt,numdp,
2 grvity,idirgv,nodspc,nspcor
      common/bk02/iburn,isdo,dt1,dt2
      common/aux13/
&zeta(nlq),thick(nlq),fga(nlq),fgb(nlq),fgc(nlq),
&gl11(nlq),gl12(nlq),gl13(nlq),gl21(nlq),gl22(nlq),gl23(nlq),
&gl31(nlq),gl32(nlq),gl33(nlq),
&x1(nlq),y1(nlq),z1(nlq),x2(nlq),y2(nlq),z2(nlq),
&x3(nlq),y3(nlq),z3(nlq),x4(nlq),y4(nlq),z4(nlq),
&fx1(nlq),fy1(nlq),fz1(nlq),fx2(nlq),fy2(nlq),fz2(nlq),
&fx3(nlq),fy3(nlq),fz3(nlq),fx4(nlq),fy4(nlq),fz4(nlq),
&xmx1(nlq),xmy1(nlq),xmz1(nlq),xmx2(nlq),xmy2(nlq),xmz2(nlq),
&xmx3(nlq),xmy3(nlq),xmz3(nlq),xmx4(nlq),xmy4(nlq),xmz4(nlq)
      common/aux33/
1 ix1(nlq),ix2(nlq),ix3(nlq),ix4(nlq),ixs(nlq,4),mxt(nlq)
      common/aux36/lft,llt

      dimension e(3,*),f(3,*),ndlist(*)
      allocatable anorm(:,:),thnod(:),nth(:)

c Set up & initialize nodal arrays
      allocate(anorm(3,numnp),thnod(numnp),nth(numnp))
      call azero(anorm,numnp*3)
      call azero(thnod,numnp)
      call iazero(nth,numnp)

c Get component normal to plane of element and add to nodes
      do 10 i=lft,llt
        anorm(1,ix1(i))=anorm(1,ix1(i))+gl13(i)
        anorm(2,ix1(i))=anorm(2,ix1(i))+gl23(i)
        anorm(3,ix1(i))=anorm(3,ix1(i))+gl33(i)
        anorm(1,ix2(i))=anorm(1,ix2(i))+gl13(i)
        anorm(2,ix2(i))=anorm(2,ix2(i))+gl23(i)
        anorm(3,ix2(i))=anorm(3,ix2(i))+gl33(i)
        anorm(1,ix3(i))=anorm(1,ix3(i))+gl13(i)
        anorm(2,ix3(i))=anorm(2,ix3(i))+gl23(i)
        anorm(3,ix3(i))=anorm(3,ix3(i))+gl33(i)
        anorm(1,ix4(i))=anorm(1,ix4(i))+gl13(i)
        anorm(2,ix4(i))=anorm(2,ix4(i))+gl23(i)
        anorm(3,ix4(i))=anorm(3,ix4(i))+gl33(i)
        thnod(ix1(i))=thnod(ix1(i))+thick(i)

```

```

        thnod(ix2(i))=thnod(ix2(i))+thick(i)
        thnod(ix3(i))=thnod(ix3(i))+thick(i)
        thnod(ix4(i))=thnod(ix4(i))+thick(i)
        nth(ix1(i))=nth(ix1(i))+1
        nth(ix2(i))=nth(ix2(i))+1
        nth(ix3(i))=nth(ix3(i))+1
        nth(ix4(i))=nth(ix4(i))+1
    10 continue

c Loop through each node
    do 20 n=1,nlstm
        i=ndlist(n)
        if(nth(i).ne.0) then
c Make each normal vector a unit vector
            amag=sqrt(anorm(1,i)**2 + anorm(2,i)**2 + anorm(3,i)**2)
            if (amag.gt.0.0) then
                anorm(1,i)=anorm(1,i)/amag
                anorm(2,i)=anorm(2,i)/amag
                anorm(3,i)=anorm(3,i)/amag
            endif
c Dot product of external moment with unit normal
            gnrm=f(1,i)*anorm(1,i)+f(2,i)*anorm(2,i)+f(3,i)*anorm(3,i)
c Dot product of external force with unit normal
            fnrm=e(1,i)*anorm(1,i)+e(2,i)*anorm(2,i)+e(3,i)*anorm(3,i)
c If gnrm is nonzero, this node as been treated before
            if(abs(gnrm).lt.0.01) then
c If not zero, add h/2 * fnrm to drilling moment
                if(abs(fnrm).gt.0.0001) then
                    fdrill=fnrm*0.5*thnod(i)/nth(i)
                    f(1,i)=anorm(1,i)*fdrill
                    f(2,i)=anorm(2,i)*fdrill
                    f(3,i)=anorm(3,i)*fdrill
                endif
            endif
        endif
    20 continue

c Release Nodal Arrays
    deallocate(anorm,thnod,nth)

    return
end

```

```

      subroutine kmfrc(e,f,qs,mte,ibls,sf1,sf2,sf3,sf4,sf5,sf6,
1          ndlist,nlstm,ym,rotall,x,x0)
c *****
c Calculates Hourglass force, adds to nodal forces from Kwon-McDermott
c Shell Element, and places in system force matrix
c (Note: sf1 through sf6 are already in global coordinates)
c McDermott 1999
c *****
      implicit double precision (a-h,o-z)
      dp
      include 'nlqpar.inc'
      common/bk00/numnp,numnp,neq,ndof,nlcur,numcl,numvc,
1      ndtpts,nelmd,nmmat,numelh,numelb,numels,numelt,numdp,
2      grvity,idirgv,nodspc,nspcor
      common/bk02/iburn,iso,dt1,dt2
      common/bk12/b12,b2,qhg
      common/aux01/
&ft11(nlq),ft12(nlq),ft13(nlq),ft21(nlq),ft22(nlq),ft23(nlq),
&fm11(nlq),fm12(nlq),fm21(nlq),fm22(nlq),
&fm31(nlq),fm32(nlq),fm41(nlq),fm42(nlq),
&fmr11(nlq),fmr12(nlq),fmr21(nlq),fmr22(nlq),fmr31(nlq),
&fmr32(nlq),fmr41(nlq),fmr42(nlq),sg5(nlq),sg6(nlq)
      common/aux7/
1      vx1(nlq),vx2(nlq),vx3(nlq),vx4(nlq),
2      vx5(nlq),vx6(nlq),vx7(nlq),vx8(nlq),
3      vy1(nlq),vy2(nlq),vy3(nlq),vy4(nlq),
4      vy5(nlq),vy6(nlq),vy7(nlq),vy8(nlq),
5      vz1(nlq),vz2(nlq),vz3(nlq),vz4(nlq),
6      vz5(nlq),vz6(nlq),vz7(nlq),vz8(nlq)
      common/aux10/area(nlq),
1      px1(nlq),px2(nlq),px3(nlq),px4(nlq),
& px5(nlq),px6(nlq),px7(nlq),px8(nlq),
2      py1(nlq),py2(nlq),py3(nlq),py4(nlq),
& py5(nlq),py6(nlq),py7(nlq),py8(nlq),
3      pz1(nlq),pz2(nlq),pz3(nlq),pz4(nlq),
& pz5(nlq),pz6(nlq),pz7(nlq),pz8(nlq),
4      dx1(nlq),dx2(nlq),dx3(nlq),dx4(nlq),
5      dx5(nlq),dx6(nlq),dx7(nlq),dx8(nlq),
6      dy1(nlq),dy2(nlq),dy3(nlq),dy4(nlq),
7      dy5(nlq),dy6(nlq),dy7(nlq),dy8(nlq),
8      dz1(nlq),dz2(nlq),dz3(nlq),dz4(nlq),
9      dz5(nlq),dz6(nlq),dz7(nlq),dz8(nlq)
      common/aux11/
&ft31(nlq),ft32(nlq),ft33(nlq),ft41(nlq),ft42(nlq),ft43(nlq),
&htx(nlq),hty(nlq),gm1(nlq),gm2(nlq),gm3(nlq),gm4(nlq),
&bsum(nlq),qhx(nlq),qhy(nlq),qwz(nlq),qtx(nlq),qty(nlq)
      common/aux13/
&zeta(nlq),thick(nlq),fga(nlq),fgb(nlq),fgc(nlq),
&gl11(nlq),gl12(nlq),gl13(nlq),gl21(nlq),gl22(nlq),gl23(nlq),
&gl31(nlq),gl32(nlq),gl33(nlq),
&x1(nlq),y1(nlq),z1(nlq),x2(nlq),y2(nlq),z2(nlq),
&x3(nlq),y3(nlq),z3(nlq),x4(nlq),y4(nlq),z4(nlq),
&fx1(nlq),fy1(nlq),fz1(nlq),fx2(nlq),fy2(nlq),fz2(nlq),
&fx3(nlq),fy3(nlq),fz3(nlq),fx4(nlq),fy4(nlq),fz4(nlq),
&xmx1(nlq),xmy1(nlq),xmz1(nlq),xmx2(nlq),xmy2(nlq),xmz2(nlq),
&xmx3(nlq),xmy3(nlq),xmz3(nlq),xmx4(nlq),xmy4(nlq),xmz4(nlq)
      common/aux33/
1      ix1(nlq),ix2(nlq),ix3(nlq),ix4(nlq),ixs(nlq,4),mxt(nlq)
      common/aux35/rhoa(nlq),cxx(nlq),fcl(nlq),fcq(nlq)

```

```

common/aux36/1ft,1lt
common/aux40/
1 x31(nlq),y31(nlq),z31(nlq),x42(nlq),y42(nlq),z42(nlq),
2 x21(nlq),y21(nlq),z21(nlq),c1(nlq),c2(nlq),c3(nlq),xl(nlq),
3 x41(nlq),y41(nlq),z41(nlq)
common/hourg/ymod,gmod,ifsv
common/sand1/ihf,ibemf,ishlf,itshf
common/sound/sndspd,sndsp(nlq),diagm(nlq),sarea(nlq),dxl(nlq)
common/ssbsis/h(8,5,5),pr(8,5,5),ps(8,5,5),pt(8,5,5),ipt,
1 nip,wgts(5,5),zet(5,5)
common/presc/voltot(nlq)
common/failu1/sieu(nlq),fail(nlq)
logical output
common/csforc/csdinc,csdout,output,ncs1,ncs2,ncs3,ncs4,ncs5,ncs6,
1 ncs7,ncs8,ncs9,numcsd
common/csfsav/savfrc(nlq,12),svfail(nlq),ndf,ifail
common/sorter/nnc,lczc,
& ns11,ns12,ns13,ns14,ns15,ns16,ns17,
& nh11,nh12,nh13,nh14,nh15,nh16,nh17,
& nt11,nt12,nt13,nt14,nt15,nt16,nt17,
& nb11,nb12,nb13,nb14,nb15,nb16,nb17

dimension e(3,*),f(3,*),qs(9,*),iblks(*),sf1(nlq,4),sf2(nlq,4),
1 sf3(nlq,4),sf4(nlq,4),sf5(nlq,4),sf6(nlq,4),ndlist(*),
2
edispg(24),eforceh(24),rotall(6,6,nlq),shapel(3),x(3,*),
3
shapef(4),derivl(2,4),derivg(3,4),derilg(3,4),bmtx(6,24),
4
bmtxt(24,6),estrain(6),estrainp(6),estress(6),estressg(6),
5
eforcehg(24,nlq),fm13(nlq),fm23(nlq),v1(3),v2(3),v3(3),
6
fm33(nlq),fm43(nlq),aj(3,3),ajinv(3,3),hour(nlq),x0(3,*),
7
edisp(24),t1(3,3),tlinv(3,3),xc(4),yc(4),zc(4),rot(6,6)

data ngausxh,ngausyh,ngauszh /2, 2, 1/
data sfac /0.8333333333333333/

ifail=0

c Force use of new HG method
hour1 = 0.5
method = 5

if (hour1.gt.1.e-04) then
c *** Specialized hourglass control based on Belytchko Stiffness form
c (Although slower, highly recommended for KM element)
if (method.eq.5) then
c Parameter setup
ngaussh=ngausxh*ngausyh*ngauszh
if (gmod.ne.0.0) then
pois = ymod / (2.0 * gmod) - 1.0
else
pois = 0.3
gmod = ymod / 2.6
endif
front = ymod / (1.0 - pois**2)

```



```

c Begin Element Loop
do 900 ie=lft,llt
    hourl(ie) = hourl * thick(ie)**2 / area(ie)

c Retrieve coordinates
    xc(1)=x0(1,ix1(ie))
    yc(1)=x0(2,ix1(ie))
    zc(1)=x0(3,ix1(ie))
    xc(2)=x0(1,ix2(ie))
    yc(2)=x0(2,ix2(ie))
    zc(2)=x0(3,ix2(ie))
    xc(3)=x0(1,ix3(ie))
    yc(3)=x0(2,ix3(ie))
    zc(3)=x0(3,ix3(ie))
    xc(4)=x0(1,ix4(ie))
    yc(4)=x0(2,ix4(ie))
    zc(4)=x0(3,ix4(ie))

c Retrieve displacements
    edisp(1) = x(1,ix1(ie))-xc(1)
    edisp(2) = x(2,ix1(ie))-yc(1)
    edisp(3) = x(3,ix1(ie))-zc(1)
    edisp(7) = x(1,ix2(ie))-xc(2)
    edisp(8) = x(2,ix2(ie))-yc(2)
    edisp(9) = x(3,ix2(ie))-zc(2)
    edisp(13) = x(1,ix3(ie))-xc(3)
    edisp(14) = x(2,ix3(ie))-yc(3)
    edisp(15) = x(3,ix3(ie))-zc(3)
    edisp(19) = x(1,ix4(ie))-xc(4)
    edisp(20) = x(2,ix4(ie))-yc(4)
    edisp(21) = x(3,ix4(ie))-zc(4)

c Retrieve Rotations (Assumes all initial rotations are zero)
    edisp(4) = x(1,numnp+ix1(ie))
    edisp(5) = x(2,numnp+ix1(ie))
    edisp(6) = x(3,numnp+ix1(ie))
    edisp(10) = x(1,numnp+ix2(ie))
    edisp(11) = x(2,numnp+ix2(ie))
    edisp(12) = x(3,numnp+ix2(ie))
    edisp(16) = x(1,numnp+ix3(ie))
    edisp(17) = x(2,numnp+ix3(ie))
    edisp(18) = x(3,numnp+ix3(ie))
    edisp(22) = x(1,numnp+ix4(ie))
    edisp(23) = x(2,numnp+ix4(ie))
    edisp(24) = x(3,numnp+ix4(ie))

c Create Transformation Matrix and get inverse
    t1(1,1)=gl11(ie)
    t1(2,1)=gl21(ie)
    t1(3,1)=gl31(ie)
    t1(1,2)=gl12(ie)
    t1(2,2)=gl22(ie)
    t1(3,2)=gl32(ie)
    t1(1,3)=gl13(ie)
    t1(2,3)=gl23(ie)
    t1(3,3)=gl33(ie)

    call kminv3(t1,tlinv,det)

c Rotate displacements to local coordinate system

```

```

do 950 i=1,4
  do 950 j=1,3
    edisp((i-1)*6+j)=edisp((i-1)*6+j)
    edisp((i-1)*6+3+j)=(edisp((i-1)*6+4)*tlinv(j,1) +
1      edisp((i-1)*6+5)*tlinv(j,2) +
2      edisp((i-1)*6+6)*tlinv(j,3))
950 continue

c Rename Direction Vectors to allow simple porting of previously
c written code
v1(1) = gl11(ie)
v1(2) = gl21(ie)
v1(3) = gl31(ie)
v2(1) = gl12(ie)
v2(2) = gl22(ie)
v2(3) = gl32(ie)
v3(1) = gl13(ie)
v3(2) = gl23(ie)
v3(3) = gl33(ie)
hzdt=thick(ie)*0.5

c Initialize Hourglass force vector
do 901 i=1,24
901   eforceh(i) = 0.0

c *** Integration Loop
do 910 ix=1,ngausxh
  rc=zet(ix,ngausxh)
  wx=wgts(ix,ngausxh)
  do 910 iy=1,ngausyh
    sc=zet(iy,ngausyh)
    wy=wgts(iy,ngausyh)
    do 910 iz=1,ngauszh
      tc=zet(iz,ngauszh)
      wz=wgts(iz,ngauszh)

c 1-D Shape Function
      call kmshap(tc,shapel)

c 2-D Shape Functions
      shapef(1)=0.25*(1.0-rc)*(1.0-sc)
      shapef(2)=0.25*(1.0+rc)*(1.0-sc)
      shapef(3)=0.25*(1.0+rc)*(1.0+sc)
      shapef(4)=0.25*(1.0-rc)*(1.0+sc)

c 2-D Shape Function Derivatives
      derivl(1,1)=-0.25*(1.0-sc)
      derivl(1,2)=0.25*(1.0-sc)
      derivl(1,3)=0.25*(1.0+sc)
      derivl(1,4)=-0.25*(1.0+sc)
      derivl(2,1)=-0.25*(1.0-rc)
      derivl(2,2)=-0.25*(1.0+rc)
      derivl(2,3)=0.25*(1.0+rc)
      derivl(2,4)=0.25*(1.0-rc)

c Compute jacobian matrix and its inverse
      hz=thick(ie)*0.5*(shapel(2)-shapel(1))

      aj(1,1)=derivl(1,1)*xc(1)+derivl(1,2)*xc(2)+derivl(1,3)*xc(3)+

```

```

1      derivl(1,4)*xc(4)+derivl(1,1)*hz*v3(1)+
2      derivl(1,2)*hz*v3(1)+derivl(1,3)*hz*v3(1)+
3      derivl(1,4)*hz*v3(1)
  aj(2,1)=derivl(2,1)*xc(1)+derivl(2,2)*xc(2)+derivl(2,3)*xc(3)+
1      derivl(2,4)*xc(4)+derivl(2,1)*hz*v3(1)+
2      derivl(2,2)*hz*v3(1)+derivl(2,3)*hz*v3(1)+
3      derivl(2,4)*hz*v3(1)
    aj(3,1)=hzdt*v3(1)
  aj(1,2)=derivl(1,1)*yc(1)+derivl(1,2)*yc(2)+derivl(1,3)*yc(3)+
1      derivl(1,4)*yc(4)+derivl(1,1)*hz*v3(2)+
2      derivl(1,2)*hz*v3(2)+derivl(1,3)*hz*v3(2)+
3      derivl(1,4)*hz*v3(2)
  aj(2,2)=derivl(2,1)*yc(1)+derivl(2,2)*yc(2)+derivl(2,3)*yc(3)+
1      derivl(2,4)*yc(4)+derivl(2,1)*hz*v3(2)+
2      derivl(2,2)*hz*v3(2)+derivl(2,3)*hz*v3(2)+
3      derivl(2,4)*hz*v3(2)
    aj(3,2)=hzdt*v3(2)
  aj(1,3)=derivl(1,1)*zc(1)+derivl(1,2)*zc(2)+derivl(1,3)*zc(3)+
1      derivl(1,4)*zc(4)+derivl(1,1)*hz*v3(3)+
2      derivl(1,2)*hz*v3(3)+derivl(1,3)*hz*v3(3)+
3      derivl(1,4)*hz*v3(3)
  aj(2,3)=derivl(2,1)*zc(1)+derivl(2,2)*zc(2)+derivl(2,3)*zc(3)+
1      derivl(2,4)*zc(4)+derivl(2,1)*hz*v3(3)+
2      derivl(2,2)*hz*v3(3)+derivl(2,3)*hz*v3(3)+
3      derivl(2,4)*hz*v3(3)
    aj(3,3)=hzdt*v3(3)

```

c

```

  call kminv3 (aj,ajinv,det)
  detwt = det * wx * wy * wz

```

c Compute global derivatives and strain-nodal displacement matrix

```

  do 902 i=1,4
    derivg(1,i)=ajinv(1,1)*derivl(1,i)+ajinv(1,2)*derivl(2,i)
    derivg(2,i)=ajinv(2,1)*derivl(1,i)+ajinv(2,2)*derivl(2,i)
    derivg(3,i)=ajinv(3,1)*derivl(1,i)+ajinv(3,2)*derivl(2,i)
    derilg(1,i)=ajinv(1,3)*hzdt
    derilg(2,i)=ajinv(2,3)*hzdt
    derilg(3,i)=ajinv(3,3)*hzdt
902  continue

  do 903 i=1,6
    do 903 j=1,24
      bmtx(i,j)=0.0
903  continue

  do 904 i=1,4
    i1=(i-1)*6+1
    i2=i1+1
    i3=i2+1
    i4=i3+1
    i5=i4+1
    i6=i5+1
    gk1=derivg(1,i)*hz+shapef(i)*derilg(1,i)
    gk2=derivg(2,i)*hz+shapef(i)*derilg(2,i)
    gk3=derivg(3,i)*hz+shapef(i)*derilg(3,i)

c
    bmtx(1,i1)=derivg(1,i)
    bmtx(1,i4)=gk1*(-v2(1))
    bmtx(1,i5)=gk1*v1(1)
    bmtx(1,i6)=gk1*v3(1)

```

```

        bmtx(2,i2)=derivg(2,i)
        bmtx(2,i4)=gk2*(-v2(2))
        bmtx(2,i5)=gk2*v1(2)
        bmtx(2,i6)=gk2*v3(2)
        bmtx(3,i3)=derivg(3,i)
        bmtx(3,i4)=gk3*(-v2(3))
        bmtx(3,i5)=gk3*v1(3)
        bmtx(3,i6)=gk3*v3(3)
        bmtx(4,i1)=derivg(2,i)
        bmtx(4,i2)=derivg(1,i)
        bmtx(4,i4)=gk2*(-v2(1))+gk1*(-v2(2))
        bmtx(4,i5)=gk2*v1(1)+gk1*v1(2)
        bmtx(4,i6)=gk2*v3(1)+gk1*v3(2)
        bmtx(5,i2)=derivg(3,i)
        bmtx(5,i3)=derivg(2,i)
        bmtx(5,i4)=gk3*(-v2(2))+gk2*(-v2(3))
        bmtx(5,i5)=gk3*v1(2)+gk2*v1(3)
        bmtx(5,i6)=gk3*v3(2)+gk2*v3(3)
        bmtx(6,i1)=derivg(3,i)
        bmtx(6,i3)=derivg(1,i)
        bmtx(6,i4)=gk3*(-v2(1))+gk1*(-v2(3))
        bmtx(6,i5)=gk3*v1(1)+gk1*v1(3)
        bmtx(6,i6)=gk3*v3(1)+gk1*v3(3)
904      continue

      do 905 i=1,6
        do 905 j=1,24
905          bmtxt(j,i)=bmtx(i,j)

c Calculate strain
      do 906 i=1,6
        estrainp(i)=0.0
        do 906 j=1,24
          estrainp(i)=estrainp(i)+bmtx(i,j)*edisp(j)
906      continue

      do 920 i=1,6
        do 920 j=1,6
          rot(i,j)=rotall(i,j,ie)
920      continue

c Transform strain to local coordinates system
      do 907 i=1,6
        estrain(i)=0.0
        do 907 j=1,6
          estrain(i)=estrain(i)+rot(i,j)*estrainp(i)
907      continue

c Calculate stress using plane-strain formulas
      estress(1) = front*(estrain(1) + pois*estrain(2))
      estress(2) = front*(pois*estrain(1) + estrain(2))
      estress(3) = ymod * estrain(3)
      estress(4) = gmod * estrain(4)
      estress(5) = sfac * gmod * estrain(5)
      estress(6) = sfac * gmod * estrain(6)

c Rotate stresses to global coordinates
      do 908 i=1,6
        estressg(i)=0.0
        do 908 j=1,6

```

```

908      estressg(i)=estressg(i)+rot(j,i)*estress(j)

c Find eforceh for this gauss point and sum to element force
      do 909 i=1,24
        do 909 j=1,6
909          eforceh(i)=eforceh(i)+bmtxt(i,j)*estressg(j)*detwt

c *** End of Integration Loop
910      continue

c Store force into HG Variables
      ft11(ie)=eforceh(1)
      ft12(ie)=eforceh(2)
      ft13(ie)=eforceh(3)
      fm11(ie)=eforceh(4)
      fm12(ie)=eforceh(5)
      fm13(ie)=eforceh(6)
      ft21(ie)=eforceh(7)
      ft22(ie)=eforceh(8)
      ft23(ie)=eforceh(9)
      fm21(ie)=eforceh(10)
      fm22(ie)=eforceh(11)
      fm23(ie)=eforceh(12)
      ft31(ie)=eforceh(13)
      ft32(ie)=eforceh(14)
      ft33(ie)=eforceh(15)
      fm31(ie)=eforceh(16)
      fm32(ie)=eforceh(17)
      fm33(ie)=eforceh(18)
      ft41(ie)=eforceh(19)
      ft42(ie)=eforceh(20)
      ft43(ie)=eforceh(21)
      fm41(ie)=eforceh(22)
      fm42(ie)=eforceh(23)
      fm43(ie)=eforceh(24)

c *** End of Element Loop
900      continue

c Other methods
      else
        do 200 i=lft,llt
          fm13(i)=0.0
          fm23(i)=0.0
          fm33(i)=0.0
          fm43(i)=0.0
200      continue

c Traditional Hour Glass Force Calculation
      tmode=qhg*ygmod/1920.0
      wmode=qhg*gmmod/120.00
      xmmode=qhg*ygmod/80.000
      do 10 i=lft,llt
        x2(i) =gl11(i)*x21(i) +gl21(i)*y21(i) +gl31(i)*z21(i)
        y2(i) =gl12(i)*x21(i) +gl22(i)*y21(i) +gl32(i)*z21(i)
        x3(i) =gl11(i)*x31(i) +gl21(i)*y31(i) +gl31(i)*z31(i)
        y3(i) =gl12(i)*x31(i) +gl22(i)*y31(i) +gl32(i)*z31(i)
        x4(i) =gl11(i)*x41(i) +gl21(i)*y41(i) +gl31(i)*z41(i)
        y4(i) =gl12(i)*x41(i) +gl22(i)*y41(i) +gl32(i)*z41(i)
        htx(i)=area(i)*(x3(i)-x2(i)-x4(i))
        hty(i)=area(i)*(y3(i)-y2(i)-y4(i))

```

```

gm1(i)= 1.-px1(i)*htx(i)-py1(i)*hty(i)
gm2(i)=-1.-px2(i)*htx(i)-py2(i)*hty(i)
gm3(i)= 2.-gm1(i)
gm4(i)=-2.-gm2(i)
10 continue
if (ifsv.eq.3) then
do 20 i=lft,llt
bsum(i)=2.*(px1(i)**2+px2(i)**2+py1(i)**2+py2(i)**2)
xl(i)=area(i)*bsum(i)*thick(i)
cl(i)=xl(i)*thick(i)**2
c2(i)=wmode*cl(i)*area(i)
c3(i)=xmode*xl(i)
cl(i)=tmode*cl(i)
20 continue
else
hgfac=qhg*rhoa(lft)*sndspd/(dtl+1.e-20)
do 30 i=lft,llt
c3(i)=hgfac*sqrt(sarea(i))*thick(i)
c2(i)=c3(i)
cl(i)=.05*c3(i)*thick(i)
30 continue
endif
do 40 i=lft,llt
qhx(i)=gm1(i)*vx1(i)+gm2(i)*vx2(i)+gm3(i)*vx3(i)+gm4(i)*vx4(i)
qhy(i)=gm1(i)*vy1(i)+gm2(i)*vy2(i)+gm3(i)*vy3(i)+gm4(i)*vy4(i)
qwz(i)=gm1(i)*vz1(i)+gm2(i)*vz2(i)+gm3(i)*vz3(i)+gm4(i)*vz4(i)
qtx(i)=gm1(i)*vx5(i)+gm2(i)*vx6(i)+gm3(i)*vx7(i)+gm4(i)*vx8(i)
qty(i)=gm1(i)*vy5(i)+gm2(i)*vy6(i)+gm3(i)*vy7(i)+gm4(i)*vy8(i)
40 continue
if (ifsv.ne.3) then
do 50 i=lft,llt
qs(1,i)=0.
qs(2,i)=0.
qs(3,i)=0.
qs(4,i)=0.
qs(5,i)=0.
50 continue
endif
do 60 i=lft,llt
qs(1,i)=qs(1,i)+c3(i)*qhx(i)
qs(2,i)=qs(2,i)+c3(i)*qhy(i)
qs(3,i)=qs(3,i)+c2(i)*qwz(i)
qs(4,i)=qs(4,i)+cl(i)*qtx(i)
qs(5,i)=qs(5,i)+cl(i)*qty(i)
ft31(i)=gm3(i)*qs(1,i)
ft32(i)=gm3(i)*qs(2,i)
ft33(i)=gm3(i)*qs(3,i)
ft41(i)=gm4(i)*qs(1,i)
ft42(i)=gm4(i)*qs(2,i)
ft43(i)=gm4(i)*qs(3,i)
ft11(i)=gm1(i)*qs(1,i)
ft12(i)=gm1(i)*qs(2,i)
ft13(i)=gm1(i)*qs(3,i)
ft21(i)=gm2(i)*qs(1,i)
ft22(i)=gm2(i)*qs(2,i)
ft23(i)=gm2(i)*qs(3,i)
fm11(i)=gm1(i)*qs(4,i)
fm12(i)=gm1(i)*qs(5,i)
fm21(i)=gm2(i)*qs(4,i)
fm22(i)=gm2(i)*qs(5,i)

```

```

        fm31(i)=gm3(i)*qs(4,i)
        fm32(i)=gm3(i)*qs(5,i)
        fm41(i)=gm4(i)*qs(4,i)
        fm42(i)=gm4(i)*qs(5,i)
60    continue
    endif
    else

c *** No Hourglass Control
    do 70 i=lft,llt
        ft31(i)=0.0
        ft32(i)=0.0
        ft33(i)=0.0
        ft41(i)=0.0
        ft42(i)=0.0
        ft43(i)=0.0
        ft11(i)=0.0
        ft12(i)=0.0
        ft13(i)=0.0
        ft21(i)=0.0
        ft22(i)=0.0
        ft23(i)=0.0
        fm11(i)=0.0
        fm12(i)=0.0
        fm13(i)=0.0
        fm21(i)=0.0
        fm22(i)=0.0
        fm23(i)=0.0
        fm31(i)=0.0
        fm32(i)=0.0
        fm33(i)=0.0
        fm41(i)=0.0
        fm42(i)=0.0
        fm43(i)=0.0
70    continue
    endif

c Total Element forces on its nodes
    if(method.ne.5) then
        do 80 i=lft,llt
            fx1(i)=gl11(i)*ft11(i)+gl12(i)*ft12(i)+gl13(i)*ft13(i)+sf1(i,1)
            fy1(i)=gl21(i)*ft11(i)+gl22(i)*ft12(i)+gl23(i)*ft13(i)+sf2(i,1)
            fz1(i)=gl31(i)*ft11(i)+gl32(i)*ft12(i)+gl33(i)*ft13(i)+sf3(i,1)
            fx2(i)=gl11(i)*ft21(i)+gl12(i)*ft22(i)+gl13(i)*ft23(i)+sf1(i,2)
            fy2(i)=gl21(i)*ft21(i)+gl22(i)*ft22(i)+gl23(i)*ft23(i)+sf2(i,2)
            fz2(i)=gl31(i)*ft21(i)+gl32(i)*ft22(i)+gl33(i)*ft23(i)+sf3(i,2)
            fx3(i)=gl11(i)*ft31(i)+gl12(i)*ft32(i)+gl13(i)*ft33(i)+sf1(i,3)
            fy3(i)=gl21(i)*ft31(i)+gl22(i)*ft32(i)+gl23(i)*ft33(i)+sf2(i,3)
            fz3(i)=gl31(i)*ft31(i)+gl32(i)*ft32(i)+gl33(i)*ft33(i)+sf3(i,3)
            fx4(i)=gl11(i)*ft41(i)+gl12(i)*ft42(i)+gl13(i)*ft43(i)+sf1(i,4)
            fy4(i)=gl21(i)*ft41(i)+gl22(i)*ft42(i)+gl23(i)*ft43(i)+sf2(i,4)
            fz4(i)=gl31(i)*ft41(i)+gl32(i)*ft42(i)+gl33(i)*ft43(i)+sf3(i,4)
            xmx1(i)=gl11(i)*fm11(i)+gl12(i)*fm12(i)+gl13(i)*fm13(i)+sf4(i,1)
            xmy1(i)=gl21(i)*fm11(i)+gl22(i)*fm12(i)+gl23(i)*fm13(i)+sf5(i,1)
            x mz1(i)=gl31(i)*fm11(i)+gl32(i)*fm12(i)+gl33(i)*fm13(i)+sf6(i,1)
            xmx2(i)=gl11(i)*fm21(i)+gl12(i)*fm22(i)+gl13(i)*fm23(i)+sf4(i,2)
            xmy2(i)=gl21(i)*fm21(i)+gl22(i)*fm22(i)+gl23(i)*fm23(i)+sf5(i,2)
            x mz2(i)=gl31(i)*fm21(i)+gl32(i)*fm22(i)+gl33(i)*fm23(i)+sf6(i,2)
            xmx3(i)=gl11(i)*fm31(i)+gl12(i)*fm32(i)+gl13(i)*fm33(i)+sf4(i,3)
            xmy3(i)=gl21(i)*fm31(i)+gl22(i)*fm32(i)+gl23(i)*fm33(i)+sf5(i,3)

```

```

      xmx3(i)=gl31(i)*fm31(i)+gl32(i)*fm32(i)+gl33(i)*fm33(i)+sf6(i,3)
      xmx4(i)=gl11(i)*fm41(i)+gl12(i)*fm42(i)+gl13(i)*fm43(i)+sf4(i,4)
      xmy4(i)=gl21(i)*fm41(i)+gl22(i)*fm42(i)+gl23(i)*fm43(i)+sf5(i,4)
      xmx4(i)=gl31(i)*fm41(i)+gl32(i)*fm42(i)+gl33(i)*fm43(i)+sf6(i,4)
80  continue
    else
      do 400 i=lft,llt
        fx1(i)=gl11(i)*ft11(i)+gl12(i)*ft12(i)+gl13(i)*ft13(i)
        fy1(i)=gl21(i)*ft11(i)+gl22(i)*ft12(i)+gl23(i)*ft13(i)
        fz1(i)=gl31(i)*ft11(i)+gl32(i)*ft12(i)+gl33(i)*ft13(i)
        fx2(i)=gl11(i)*ft21(i)+gl12(i)*ft22(i)+gl13(i)*ft23(i)
        fy2(i)=gl21(i)*ft21(i)+gl22(i)*ft22(i)+gl23(i)*ft23(i)
        fz2(i)=gl31(i)*ft21(i)+gl32(i)*ft22(i)+gl33(i)*ft23(i)
        fx3(i)=gl11(i)*ft31(i)+gl12(i)*ft32(i)+gl13(i)*ft33(i)
        fy3(i)=gl21(i)*ft31(i)+gl22(i)*ft32(i)+gl23(i)*ft33(i)
        fz3(i)=gl31(i)*ft31(i)+gl32(i)*ft32(i)+gl33(i)*ft33(i)
        fx4(i)=gl11(i)*ft41(i)+gl12(i)*ft42(i)+gl13(i)*ft43(i)
        fy4(i)=gl21(i)*ft41(i)+gl22(i)*ft42(i)+gl23(i)*ft43(i)
        fz4(i)=gl31(i)*ft41(i)+gl32(i)*ft42(i)+gl33(i)*ft43(i)
        xmx1(i)=gl11(i)*fm11(i)+gl12(i)*fm12(i)+gl13(i)*fm13(i)
        xmy1(i)=gl21(i)*fm11(i)+gl22(i)*fm12(i)+gl23(i)*fm13(i)
        xmx1(i)=gl31(i)*fm11(i)+gl32(i)*fm12(i)+gl33(i)*fm13(i)
        xmx2(i)=gl11(i)*fm21(i)+gl12(i)*fm22(i)+gl13(i)*fm23(i)
        xmy2(i)=gl21(i)*fm21(i)+gl22(i)*fm22(i)+gl23(i)*fm23(i)
        xmx2(i)=gl31(i)*fm21(i)+gl32(i)*fm22(i)+gl33(i)*fm23(i)
        xmx3(i)=gl11(i)*fm31(i)+gl12(i)*fm32(i)+gl13(i)*fm33(i)
        xmy3(i)=gl21(i)*fm31(i)+gl22(i)*fm32(i)+gl23(i)*fm33(i)
        xmx3(i)=gl31(i)*fm31(i)+gl32(i)*fm32(i)+gl33(i)*fm33(i)
        xmx4(i)=gl11(i)*fm41(i)+gl12(i)*fm42(i)+gl13(i)*fm43(i)
        xmy4(i)=gl21(i)*fm41(i)+gl22(i)*fm42(i)+gl23(i)*fm43(i)
        xmx4(i)=gl31(i)*fm41(i)+gl32(i)*fm42(i)+gl33(i)*fm43(i)
400  continue
      do 405 i=lft,llt
        fx1(i)=sf1(i,1)+hour(i)*(fx1(i)-sf1(i,1))
        fy1(i)=sf2(i,1)+hour(i)*(fy1(i)-sf2(i,1))
        fz1(i)=sf3(i,1)+hour(i)*(fz1(i)-sf3(i,1))
        fx2(i)=sf1(i,2)+hour(i)*(fx2(i)-sf1(i,2))
        fy2(i)=sf2(i,2)+hour(i)*(fy2(i)-sf2(i,2))
        fz2(i)=sf3(i,2)+hour(i)*(fz2(i)-sf3(i,2))
        fx3(i)=sf1(i,3)+hour(i)*(fx3(i)-sf1(i,3))
        fy3(i)=sf2(i,3)+hour(i)*(fy3(i)-sf2(i,3))
        fz3(i)=sf3(i,3)+hour(i)*(fz3(i)-sf3(i,3))
        fx4(i)=sf1(i,4)+hour(i)*(fx4(i)-sf1(i,4))
        fy4(i)=sf2(i,4)+hour(i)*(fy4(i)-sf2(i,4))
        fz4(i)=sf3(i,4)+hour(i)*(fz4(i)-sf3(i,4))
        xmx1(i)=sf4(i,1)+hour(i)*(xmx1(i)-sf4(i,1))
        xmy1(i)=sf5(i,1)+hour(i)*(xmy1(i)-sf5(i,1))
        xmx1(i)=sf6(i,1)+hour(i)*(xmx1(i)-sf6(i,1))
        xmx2(i)=sf4(i,2)+hour(i)*(xmx2(i)-sf4(i,2))
        xmy2(i)=sf5(i,2)+hour(i)*(xmy2(i)-sf5(i,2))
        xmx2(i)=sf6(i,2)+hour(i)*(xmx2(i)-sf6(i,2))
        xmx3(i)=sf4(i,3)+hour(i)*(xmx3(i)-sf4(i,3))
        xmy3(i)=sf5(i,3)+hour(i)*(xmy3(i)-sf5(i,3))
        xmx3(i)=sf6(i,3)+hour(i)*(xmx3(i)-sf6(i,3))
        xmx4(i)=sf4(i,4)+hour(i)*(xmx4(i)-sf4(i,4))
        xmy4(i)=sf5(i,4)+hour(i)*(xmy4(i)-sf5(i,4))
        xmx4(i)=sf6(i,4)+hour(i)*(xmx4(i)-sf6(i,4))
405  continue
      endif

```



```

c Apply drilling moments for contact forces
  call kmdrill(e,f,ndlist,nlstm)

```

```

c Put nodal forces into system force vector
  if (ishlf.ne.1) then

```

```

c    No element failure
    do 90 i=lft,llt
      e(1,ix1(i))=e(1,ix1(i))-fx1(i)
      e(2,ix1(i))=e(2,ix1(i))-fy1(i)
      e(3,ix1(i))=e(3,ix1(i))-fz1(i)
      f(1,ix1(i))=f(1,ix1(i))-xmx1(i)
      f(2,ix1(i))=f(2,ix1(i))-xmy1(i)
      f(3,ix1(i))=f(3,ix1(i))-xmz1(i)
      e(1,ix2(i))=e(1,ix2(i))-fx2(i)
      e(2,ix2(i))=e(2,ix2(i))-fy2(i)
      e(3,ix2(i))=e(3,ix2(i))-fz2(i)
      f(1,ix2(i))=f(1,ix2(i))-xmx2(i)
      f(2,ix2(i))=f(2,ix2(i))-xmy2(i)
      f(3,ix2(i))=f(3,ix2(i))-xmz2(i)
      e(1,ix3(i))=e(1,ix3(i))-fx3(i)
      e(2,ix3(i))=e(2,ix3(i))-fy3(i)
      e(3,ix3(i))=e(3,ix3(i))-fz3(i)
      f(1,ix3(i))=f(1,ix3(i))-xmx3(i)
      f(2,ix3(i))=f(2,ix3(i))-xmy3(i)
      f(3,ix3(i))=f(3,ix3(i))-xmz3(i)
      e(1,ix4(i))=e(1,ix4(i))-fx4(i)
      e(2,ix4(i))=e(2,ix4(i))-fy4(i)
      e(3,ix4(i))=e(3,ix4(i))-fz4(i)
      f(1,ix4(i))=f(1,ix4(i))-xmx4(i)
      f(2,ix4(i))=f(2,ix4(i))-xmy4(i)
      f(3,ix4(i))=f(3,ix4(i))-xmz4(i)
90    continue

```

```

  else

```

```

c    With Element Failure
    ifail=1
    do 100 i=lft,llt
      e(1,ix1(i))=e(1,ix1(i))-fail(i)*fx1(i)
      e(2,ix1(i))=e(2,ix1(i))-fail(i)*fy1(i)
      e(3,ix1(i))=e(3,ix1(i))-fail(i)*fz1(i)
      f(1,ix1(i))=f(1,ix1(i))-fail(i)*xmx1(i)
      f(2,ix1(i))=f(2,ix1(i))-fail(i)*xmy1(i)
      f(3,ix1(i))=f(3,ix1(i))-fail(i)*xmz1(i)
      e(1,ix2(i))=e(1,ix2(i))-fail(i)*fx2(i)
      e(2,ix2(i))=e(2,ix2(i))-fail(i)*fy2(i)
      e(3,ix2(i))=e(3,ix2(i))-fail(i)*fz2(i)
      f(1,ix2(i))=f(1,ix2(i))-fail(i)*xmx2(i)
      f(2,ix2(i))=f(2,ix2(i))-fail(i)*xmy2(i)
      f(3,ix2(i))=f(3,ix2(i))-fail(i)*xmz2(i)
      e(1,ix3(i))=e(1,ix3(i))-fail(i)*fx3(i)
      e(2,ix3(i))=e(2,ix3(i))-fail(i)*fy3(i)
      e(3,ix3(i))=e(3,ix3(i))-fail(i)*fz3(i)
      f(1,ix3(i))=f(1,ix3(i))-fail(i)*xmx3(i)
      f(2,ix3(i))=f(2,ix3(i))-fail(i)*xmy3(i)
      f(3,ix3(i))=f(3,ix3(i))-fail(i)*xmz3(i)
      e(1,ix4(i))=e(1,ix4(i))-fail(i)*fx4(i)
      e(2,ix4(i))=e(2,ix4(i))-fail(i)*fy4(i)
      e(3,ix4(i))=e(3,ix4(i))-fail(i)*fz4(i)
100   continue

```

```

        f(1,ix4(i))=f(1,ix4(i))-fail(i)*xmz4(i)
        f(2,ix4(i))=f(2,ix4(i))-fail(i)*xmy4(i)
        f(3,ix4(i))=f(3,ix4(i))-fail(i)*xmz4(i)
100  continue
    endif

    if (output) then
        nblksz=nlq
        nwords=12*nblksz
        call blkcpy(fx1,savfrc,nwords)
        ndof=4
        if (ifail.eq.1) call blkcpy (fail,svfail,nlq)
    endif
    return
end

```

```

subroutine kmtran
implicit double precision (a-h,o-z)

include 'nlqpar.inc'
c Computes B matrix based on Belytschko-Tsay shell element needed for
c hourglass control of Kwon-McDermott shell element. Also computes
c area for time-step control

common/bk02/iburn, isdo, dt1, dt2
common/aux5/
1b1vx(nlq), b1vy(nlq), b1vz(nlq), b2vx(nlq), b2vy(nlq), b2vz(nlq),
2b1tx(nlq), b1ty(nlq), b2tx(nlq), b2ty(nlq), bxyv(nlq), bxyt(nlq),
3epyz(nlq), epzx(nlq)
common/aux7/
1 vx1(nlq), vx2(nlq), vx3(nlq), vx4(nlq),
2 vx5(nlq), vx6(nlq), vx7(nlq), vx8(nlq),
3 vy1(nlq), vy2(nlq), vy3(nlq), vy4(nlq),
4 vy5(nlq), vy6(nlq), vy7(nlq), vy8(nlq),
5 vz1(nlq), vz2(nlq), vz3(nlq), vz4(nlq),
6 vz5(nlq), vz6(nlq), vz7(nlq), vz8(nlq)
common/aux10/area(nlq),
1 px1(nlq), px2(nlq), px3(nlq), px4(nlq),
& px5(nlq), px6(nlq), px7(nlq), px8(nlq),
2 py1(nlq), py2(nlq), py3(nlq), py4(nlq),
& py5(nlq), py6(nlq), py7(nlq), py8(nlq),
3 pz1(nlq), pz2(nlq), pz3(nlq), pz4(nlq),
& pz5(nlq), pz6(nlq), pz7(nlq), pz8(nlq),
4 dx1(nlq), dx2(nlq), dx3(nlq), dx4(nlq),
5 dx5(nlq), dx6(nlq), dx7(nlq), dx8(nlq),
6 dy1(nlq), dy2(nlq), dy3(nlq), dy4(nlq),
7 dy5(nlq), dy6(nlq), dy7(nlq), dy8(nlq),
8 dz1(nlq), dz2(nlq), dz3(nlq), dz4(nlq),
9 dz5(nlq), dz6(nlq), dz7(nlq), dz8(nlq)
common/aux11/
&gm11(nlq), gm12(nlq), gm13(nlq), gm21(nlq), gm22(nlq), gm23(nlq),
&gm31(nlq), gm32(nlq), gm33(nlq), px1a(nlq), py1a(nlq), px2a(nlq),
&py2a(nlq), diag1(nlq), diag2(nlq)
common/aux13/
&zeta(nlq), thick(nlq), fga(nlq), fgb(nlq), fgc(nlq),
&gl11(nlq), gl12(nlq), gl13(nlq), gl21(nlq), gl22(nlq), gl23(nlq),
&gl31(nlq), gl32(nlq), gl33(nlq),
&x1(nlq), y1(nlq), z1(nlq), x2(nlq), y2(nlq), z2(nlq),
&x3(nlq), y3(nlq), z3(nlq), x4(nlq), y4(nlq), z4(nlq)
common/aux12/
1 wxx1(nlq), wxx2(nlq), wxx3(nlq), wxx4(nlq),
2 wyy1(nlq), wyy2(nlq), wyy3(nlq), wyy4(nlq),
3 wzz1(nlq), wzz2(nlq), wzz3(nlq), wzz4(nlq),
4 a13(nlq), a23(nlq), a33(nlq)
common/aux36/lft, llt
common/aux40/
&x31(nlq), y31(nlq), z31(nlq), x42(nlq), y42(nlq), z42(nlq),
&x21(nlq), y21(nlq), z21(nlq), c1(nlq), c2(nlq), c3(nlq), x1(nlq),
&x41(nlq), y41(nlq), z41(nlq)
common/sound/sndspd, sndsp(nlq), diagm(nlq), sarea(nlq), dx1(nlq)
common/sides/sidmn(nlq)
common/double/iprec, ncpw, unit

c
dimension vx13(nlq), vx24(nlq), vy13(nlq), vy24(nlq),
1 vz13(nlq), vz24(nlq), wxx13(nlq), wxx24(nlq), wyy13(nlq),

```

```

2 wyy24(nlq),sidem(nlq),xt2(nlq),yt2(nlq),xt3(nlq),yt3(nlq),
3 xt4(nlq),yt4(nlq)
equivalence (wxx1,vx13), (wxx2,vx24), (wxx3,vy13), (wxx4,vy24),
1 (wyy1,wxx13), (wyy2,wxx24), (wyy3,wyy13), (wyy4,wyy24),
2 (wzz1,vz13), (wzz2,vz24)

```

c

```

do 10 i=lft,llt
gm11(i)=dt1*gl11(i)
gm21(i)=dt1*gl21(i)
gm31(i)=dt1*gl31(i)
gm12(i)=dt1*gl12(i)
gm22(i)=dt1*gl22(i)
gm32(i)=dt1*gl32(i)
gm13(i)=dt1*gl13(i)
gm23(i)=dt1*gl23(i)
gm33(i)=dt1*gl33(i)
vx1(i)=gm11(i)*dx1(i) +gm21(i)*dy1(i) +gm31(i)*dz1(i)
vy1(i)=gm12(i)*dx1(i) +gm22(i)*dy1(i) +gm32(i)*dz1(i)
vz1(i)=gm13(i)*dx1(i) +gm23(i)*dy1(i) +gm33(i)*dz1(i)
vx2(i)=gm11(i)*dx2(i) +gm21(i)*dy2(i) +gm31(i)*dz2(i)
vy2(i)=gm12(i)*dx2(i) +gm22(i)*dy2(i) +gm32(i)*dz2(i)
vz2(i)=gm13(i)*dx2(i) +gm23(i)*dy2(i) +gm33(i)*dz2(i)
vx3(i)=gm11(i)*dx3(i) +gm21(i)*dy3(i) +gm31(i)*dz3(i)
vy3(i)=gm12(i)*dx3(i) +gm22(i)*dy3(i) +gm32(i)*dz3(i)
vz3(i)=gm13(i)*dx3(i) +gm23(i)*dy3(i) +gm33(i)*dz3(i)
10 continue
do 20 i=lft,llt
vx4(i)=gm11(i)*dx4(i) +gm21(i)*dy4(i) +gm31(i)*dz4(i)
vy4(i)=gm12(i)*dx4(i) +gm22(i)*dy4(i) +gm32(i)*dz4(i)
vz4(i)=gm13(i)*dx4(i) +gm23(i)*dy4(i) +gm33(i)*dz4(i)
vx5(i)=gm11(i)*wxx1(i)+gm21(i)*wyy1(i)+gm31(i)*wzz1(i)
vy5(i)=gm12(i)*wxx1(i)+gm22(i)*wyy1(i)+gm32(i)*wzz1(i)
vx6(i)=gm11(i)*wxx2(i)+gm21(i)*wyy2(i)+gm31(i)*wzz2(i)
vy6(i)=gm12(i)*wxx2(i)+gm22(i)*wyy2(i)+gm32(i)*wzz2(i)
vx7(i)=gm11(i)*wxx3(i)+gm21(i)*wyy3(i)+gm31(i)*wzz3(i)
vy7(i)=gm12(i)*wxx3(i)+gm22(i)*wyy3(i)+gm32(i)*wzz3(i)
vx8(i)=gm11(i)*wxx4(i)+gm21(i)*wyy4(i)+gm31(i)*wzz4(i)
vy8(i)=gm12(i)*wxx4(i)+gm22(i)*wyy4(i)+gm32(i)*wzz4(i)
xt2(i) =gl11(i)*x21(i) +gl21(i)*y21(i) +gl31(i)*z21(i)
yt2(i) =gl12(i)*x21(i) +gl22(i)*y21(i) +gl32(i)*z21(i)
xt3(i) =gl11(i)*x31(i) +gl21(i)*y31(i) +gl31(i)*z31(i)
yt3(i) =gl12(i)*x31(i) +gl22(i)*y31(i) +gl32(i)*z31(i)
xt4(i) =gl11(i)*x41(i) +gl21(i)*y41(i) +gl31(i)*z41(i)
yt4(i) =gl12(i)*x41(i) +gl22(i)*y41(i) +gl32(i)*z41(i)
20 continue

```

c

```

do 30 i=lft,llt
px1(i) = .5*(yt2(i)-yt4(i))
px2(i) = .5* yt3(i)
py1(i) = -.5*(xt2(i)-xt4(i))
py2(i) = -.5* xt3(i)
sarea(i)=2.0*(py2(i)*px1(i)-py1(i)*px2(i))
diag1(i)=xt3(i)**2+yt3(i)**2
diag2(i)=4.*(px1(i)*px1(i)+py1(i)*py1(i))
diagm(i)= max(diag1(i),diag2(i))
side1 =xt2(i)*xt2(i)+yt2(i)*yt2(i)
side2 =(xt3(i)-xt2(i))**2+(yt3(i)-yt2(i))**2
side3 =(xt4(i)-xt3(i))**2+(yt4(i)-yt3(i))**2-1.e-14
side4 =xt4(i)*xt4(i)+yt4(i)*yt4(i)

```

```

    sida3 =side4*(5.-sign(.5*unit,side3))+side3
    sidmn(i)= min(sidel,side2,sida3,side4)
    sidem(i)= max(sidel,side2,side3,side4)*
1    (.625+sign(.375*unit,side3))
    fgb(i) =sarea(i)*fgb(i)
30 continue

```

```

c
do 40 i=lft,llt
    area(i)=1./(sarea(i)+1.e-20)
    sarea(i) = sarea(i) * sqrt(2.0*thick(i))
40 continue

```

```

c
if (isdo.eq.0.or.isdo.eq.2) then
do 50 i=lft,llt
50  diagm(i)= min(diagm(i),sidem(i))
    return
else
    return
endif

end

```

```

subroutine kwnmcd(rule,ixp,x,rhs,vt,vr,strain,yhatn,fibl,
1 auxvec,mtype,ro,cm,csprop,nsubgv,mtnum,nfegp,ihgq,hgq,xies,ener,
2 mpusr,lav,nmel,nnml,mxe,ibls,dampk,ym,prv,emain)
implicit double precision (a-h,o-z)

include 'nlqpar.inc'
c *****
c main subroutine for the kwon-mcdermott shell formulation
c (Call parameters are same as all other shell formulations)
c McDermott 1999
c *****
common/ /b(1)
common/bk00/numnp,numpc,numlp,neq,ndof,nlcur,numcl,numvc,
1 ndtpts,nelmd,mmmat,numelh,numelb,numels,numelt,numdp,
2 grvity,idirgv,nodspc,nspcor
common/bk02/iburn,isdo,dt1,dt2
common/bk12/b12,b2,qhg
common/bk13/lc0,lc1h,lc1b,lc1s,lc1t,lc2,lc3,lc4,lc5,lc6,lc7,lc9,
1 lc10,lc11,lc12,lc13,lc14,lc15,lc16,lc17,lc18,lb0,lb1,lb2,
2 lc7a,lc7b
common/bk19/nconst(60),lenma,ncneos(15)
common/bk25/dfavg,detavg,davg,iflg,ielmtc,ityptc
common/aux01/
&ft11(nlq),ft12(nlq),ft13(nlq),ft21(nlq),ft22(nlq),ft23(nlq),
&fm11(nlq),fm12(nlq),fm21(nlq),fm22(nlq),
&fm31(nlq),fm32(nlq),fm41(nlq),fm42(nlq),
&fmr11(nlq),fmr12(nlq),fmr21(nlq),fmr22(nlq),fmr31(nlq),
&fmr32(nlq),fmr41(nlq),fmr42(nlq),sg5(nlq),sg6(nlq)
common/aux2/d1(nlq),d2(nlq),d3(nlq),d4(nlq),d5(nlq),d6(nlq),
1 wzzdt(nlq),wyydt(nlq),wxxdt(nlq),einc(nlq)
common/aux7/
1 vx1(nlq),vx2(nlq),vx3(nlq),vx4(nlq),
2 vx5(nlq),vx6(nlq),vx7(nlq),vx8(nlq),
3 vy1(nlq),vy2(nlq),vy3(nlq),vy4(nlq),
4 vy5(nlq),vy6(nlq),vy7(nlq),vy8(nlq),
5 vz1(nlq),vz2(nlq),vz3(nlq),vz4(nlq),
6 vz5(nlq),vz6(nlq),vz7(nlq),vz8(nlq)
common/aux10/area(nlq),
1 px1(nlq),px2(nlq),px3(nlq),px4(nlq),
& px5(nlq),px6(nlq),px7(nlq),px8(nlq),
2 py1(nlq),py2(nlq),py3(nlq),py4(nlq),
& py5(nlq),py6(nlq),py7(nlq),py8(nlq),
3 pz1(nlq),pz2(nlq),pz3(nlq),pz4(nlq),
& pz5(nlq),pz6(nlq),pz7(nlq),pz8(nlq),
4 dx1(nlq),dx2(nlq),dx3(nlq),dx4(nlq),
5 dx5(nlq),dx6(nlq),dx7(nlq),dx8(nlq),
6 dy1(nlq),dy2(nlq),dy3(nlq),dy4(nlq),
7 dy5(nlq),dy6(nlq),dy7(nlq),dy8(nlq),
8 dz1(nlq),dz2(nlq),dz3(nlq),dz4(nlq),
9 dz5(nlq),dz6(nlq),dz7(nlq),dz8(nlq)
common/aux13/
&zeta(nlq),thick(nlq),fga(nlq),fgb(nlq),fgc(nlq),
&gl11(nlq),gl12(nlq),gl13(nlq),gl21(nlq),gl22(nlq),gl23(nlq),
&gl31(nlq),gl32(nlq),gl33(nlq),
&x1(nlq),y1(nlq),z1(nlq),x2(nlq),y2(nlq),z2(nlq),
&x3(nlq),y3(nlq),z3(nlq),x4(nlq),y4(nlq),z4(nlq)
common/aux14/
1 sig1(nlq),sig2(nlq),sig3(nlq),sig4(nlq),
2 sig5(nlq),sig6(nlq), ep(nlq),epx1(nlq),
3 epx2(nlq),epx4(nlq),epx5(nlq),epx6(nlq),aux(nlq,59)

```

```

common/aux12/
1 wxx1(nlq),wxx2(nlq),wxx3(nlq),wxx4(nlq),
2 wyy1(nlq),wyy2(nlq),wyy3(nlq),wyy4(nlq),
3 wzz1(nlq),wzz2(nlq),wzz3(nlq),wzz4(nlq),
4 a13(nlq),a23(nlq),a33(nlq)
common/aux33/
1 ix1(nlq),ix2(nlq),ix3(nlq),ix4(nlq),ixs(nlq,4),mxt(nlq)
common/aux35/rhoa(nlq),cxx(nlq),fcl(nlq),fcq(nlq)
common/aux36/lft,llt
common/sound/sndspd,sndsp(nlq),diagm(nlq),sarea(nlq),dxl(nlq)
common/bktb/ntbsl,nods,nodm,ips,ipm,ipa,ipb,ipc,ipd,
1 ipe,ipf,ipg,iph,ipi,ipj,ipk
common/ssbsis/h(8,5,5),pr(8,5,5),ps(8,5,5),pt(8,5,5),ipt,
1 nip,wgts(5,5),zet(5,5)
common/shlopt/istrn,istupd,ibelyt
common/hourg/ymod,gmod,ifsv
common/sand1/ihf,ibemf,ishlf,itshf
common/failul/sieu(nlq),fail(nlq)
common/energy/xinen
common/kinet/enkint(nlq),xmomnt(nlq),ymomnt(nlq),zmomnt(nlq)

dimension ixp(5,*),x(3,*),rhs(*),vt(3,*),vr(3,*),yhatn(12,*),
1 auxvec(*),mtype(*),ro(*),cm(*),csprop(24,*),rule(mpusr,3,*),
2 fibl(9,*),nsubgv(*),mtnum(*),nfegp(*),ihgq(*),hgq(*),
3 strain(12,*),isrn(2,6),ener(*),iblkls(*),xies(*),emain(6,nlq),
4 shapef(4),aj(3,3),ajinv(3,3),tl(3,3),tlinv(3,3),edispg(24),
5 edisp(24),shapel(3),derivg(3,4),v1(3),v2(3),v3(3),rot6t(6,6),
6 deriva(4),derivb(4),derilg(3,4),estrain(6),estrainp(6),
7 bmtx(6,24,nlq),rotall(6,6,nlq),ajdet(nlq),tlall(3,3,nlq),
8 estress(6),estressg(6),eforce(24),eforceg(24),sforcel(nlq,4),
9 sforce2(nlq,4),sforce3(nlq,4),sforce4(nlq,4),sforce5(nlq,4),
1 sforce6(nlq,4),rotrn(6,6),bmtxt(24,6),str33(nlq),ndlist(nlq*4)

data isrn/1,1,1,2,2,3,1,4,2,5,0,0/
data deriva/-0.25,0.25,0.25,-0.25/
data derivb/-0.25,-0.25,0.25,0.25/

rho=1./ro(mxe)
nip=nint(csprop(2,mxe))
irl=nint(csprop(4,mxe))
nrl=iabs(irl)
if (nip.gt.5.and.irl.eq.0) irl=1
zbr=csprop(13,mxe)
ifsv=ihgq(mxe)
ipt=1
rhoa(lft)=ro(mxe)
qhgh=.25*hgq(mxe)
mte=mtype(mxe)
if (mte.eq.20) return
nmtcon=7+nconst(mte)
sndspd=1.e-20
ym=cm(48*(mxt(lft)-1)+1)
do 10 i=lft,llt
  thick(i)=fibl(1,nnml+i)
  str33(i)=0.0
  do 10 j=1,4
    sforcel(i,j) = 0.0
    sforce2(i,j) = 0.0
    sforce3(i,j) = 0.0
    sforce4(i,j) = 0.0

```

```

        sforce5(i,j) = 0.0
        sforce6(i,j) = 0.0
10 continue

c *** Support for Drilling Moment addition to reflect pressure
c     applied to surface, which requires a list of nodes for this
c     group of elements. This list might be more efficiently generated
c     outside the time loop (in the initialize phase), but that would
c     require a lot of changes in initlz et al., plus the addition of
c     and additional global array. Using this list allows the drilling
c     moment calculations to be made
c     for only those nodes that are attached to KM shell elements,
which
c     can result in significant speed gains on large systems with
multiple
c     multiple element types.

c Create node list
do 15 i=lft,llt
    jn=(i-1)*4
    ndlist(jn+1)=ix1(i)
    ndlist(jn+2)=ix2(i)
    ndlist(jn+3)=ix3(i)
    ndlist(jn+4)=ix4(i)
15 continue

c Remove duplicates (no need to sort)
nlstm=(llt-lft+1)*4
i=1
do while (i.lt.nlstm)
    j=i+1
    do while (j.le.nlstm)
        if (ndlist(i).eq.ndlist(j)) then
            do 16 k=j,nlstm-1
                ndlist(k)=ndlist(k+1)
16         ndlist(nlstm)=0
                nlstm=nlstm-1
            endif
            j=j+1
        enddo
        i=i+1
    enddo

    if (ishlf.ne.1) then

c Retrieve Nodal Coordinates and velocities for no-failure
do 20 i=lft,llt
    sieu(i)=xies(nnml+i)
    x1(i) =x(1,ix1(i))
    y1(i) =x(2,ix1(i))
    z1(i) =x(3,ix1(i))
    x2(i) =x(1,ix2(i))
    y2(i) =x(2,ix2(i))
    z2(i) =x(3,ix2(i))
    x3(i) =x(1,ix3(i))
    y3(i) =x(2,ix3(i))
    z3(i) =x(3,ix3(i))
    x4(i) =x(1,ix4(i))
    y4(i) =x(2,ix4(i))
    z4(i) =x(3,ix4(i))

```



```

dx1(i)=vt(1,ix1(i))
dy1(i)=vt(2,ix1(i))
dz1(i)=vt(3,ix1(i))
dx2(i)=vt(1,ix2(i))
dy2(i)=vt(2,ix2(i))
dz2(i)=vt(3,ix2(i))
dx3(i)=vt(1,ix3(i))
dy3(i)=vt(2,ix3(i))
dz3(i)=vt(3,ix3(i))
dx4(i)=vt(1,ix4(i))
dy4(i)=vt(2,ix4(i))
dz4(i)=vt(3,ix4(i))
wxx1(i)=vr(1,ix1(i))
wyy1(i)=vr(2,ix1(i))
wzz1(i)=vr(3,ix1(i))
wxx2(i)=vr(1,ix2(i))
wyy2(i)=vr(2,ix2(i))
wzz2(i)=vr(3,ix2(i))
wxx3(i)=vr(1,ix3(i))
wyy3(i)=vr(2,ix3(i))
wzz3(i)=vr(3,ix3(i))
wxx4(i)=vr(1,ix4(i))
wyy4(i)=vr(2,ix4(i))
wzz4(i)=vr(3,ix4(i))
20 continue

```

```

else

```

```

c Retrieve Nodal Coordinates and Velocities w/failure

```

```

do 30 i=lft,llt
  sieu(i)=xies(nnm1+i)
  x1(i) =x(1,ix1(i))
  y1(i) =x(2,ix1(i))
  z1(i) =x(3,ix1(i))
  x2(i) =x(1,ix2(i))
  y2(i) =x(2,ix2(i))
  z2(i) =x(3,ix2(i))
  x3(i) =x(1,ix3(i))
  y3(i) =x(2,ix3(i))
  z3(i) =x(3,ix3(i))
  x4(i) =x(1,ix4(i))
  y4(i) =x(2,ix4(i))
  z4(i) =x(3,ix4(i))
  dx1(i)=vt(1,ix1(i))*fail(i)
  dy1(i)=vt(2,ix1(i))*fail(i)
  dz1(i)=vt(3,ix1(i))*fail(i)
  dx2(i)=vt(1,ix2(i))*fail(i)
  dy2(i)=vt(2,ix2(i))*fail(i)
  dz2(i)=vt(3,ix2(i))*fail(i)
  dx3(i)=vt(1,ix3(i))*fail(i)
  dy3(i)=vt(2,ix3(i))*fail(i)
  dz3(i)=vt(3,ix3(i))*fail(i)
  dx4(i)=vt(1,ix4(i))*fail(i)
  dy4(i)=vt(2,ix4(i))*fail(i)
  dz4(i)=vt(3,ix4(i))*fail(i)
  wxx1(i)=vr(1,ix1(i))*fail(i)
  wyy1(i)=vr(2,ix1(i))*fail(i)
  wzz1(i)=vr(3,ix1(i))*fail(i)
  wxx2(i)=vr(1,ix2(i))*fail(i)
  wyy2(i)=vr(2,ix2(i))*fail(i)

```

```

        wzz2(i)=vr(3,ix2(i))*fail(i)
        wx3(i)=vr(1,ix3(i))*fail(i)
        wyy3(i)=vr(2,ix3(i))*fail(i)
        wzz3(i)=vr(3,ix3(i))*fail(i)
        wx4(i)=vr(1,ix4(i))*fail(i)
        wyy4(i)=vr(2,ix4(i))*fail(i)
        wzz4(i)=vr(3,ix4(i))*fail(i)
30    continue
    endif

c Calculate Momentums and Kinetic Energy
    rho8th = 0.125*rhoa(lft)
    rho4th = 0.25*rhoa(lft)
    do 35 i=lft,llt
        dx=dx1(i)**2+dx2(i)**2+dx3(i)**2+dx4(i)**2
        dy=dy1(i)**2+dy2(i)**2+dy3(i)**2+dy4(i)**2
        dz=dz1(i)**2+dz2(i)**2+dz3(i)**2+dz4(i)**2
        enkint(i)=fibr(1,nnml+i)*rho8th*(dx+dy+dz)
        xmomnt(i)=fibr(1,nnml+i)*rho4th*(dx1(i)+dx2(i)+dx3(i)+dx4(i))
        ymomnt(i)=fibr(1,nnml+i)*rho4th*(dy1(i)+dy2(i)+dy3(i)+dy4(i))
        zmomnt(i)=fibr(1,nnml+i)*rho4th*(dz1(i)+dz2(i)+dz3(i)+dz4(i))
35    continue

c Calculate laminae vectors and surface area
    call dfnls(fibr(1,nnml+1),nip)
    call kmtran

c *** Loop through elements, get transformed displacement ***
c     and calculate rotation matrices
    do 70 ie=lft,llt

c Rename unit vectors locally to facilitate strain calculation
c     Node 1 to 2
        v1(1) = gl11(ie)
        v1(2) = gl21(ie)
        v1(3) = gl31(ie)
c     Node 1 to 4 (Actually V1 x V3)
        v2(1) = gl12(ie)
        v2(2) = gl22(ie)
        v2(3) = gl32(ie)
c     Normal to reference plane
        v3(1) = gl13(ie)
        v3(2) = gl23(ie)
        v3(3) = gl33(ie)

c Rotation transformation matrix, and its inverse
        do 40 i=1,3
            t1(i,1) = v1(i)
            t1(i,2) = v2(i)
            t1(i,3) = v3(i)
40        continue

        call kminv3(t1,tlinv,det)

c Store for use in force transformation
        do 50 i=1,3
            do 50 j=1,3
20          tlall(i,j,ie)=t1(i,j)
50        continue

c Strain Transformation Matrix

```

```

rotall(1,1,ie)=v1(1)**2
rotall(1,2,ie)=v1(2)**2
rotall(1,3,ie)=v1(3)**2
rotall(1,4,ie)=v1(1)*v1(2)
rotall(1,5,ie)=v1(2)*v1(3)
rotall(1,6,ie)=v1(1)*v1(3)
rotall(2,1,ie)=v2(1)**2
rotall(2,2,ie)=v2(2)**2
rotall(2,3,ie)=v2(3)**2
rotall(2,4,ie)=v2(1)*v2(2)
rotall(2,5,ie)=v2(2)*v2(3)
rotall(2,6,ie)=v2(1)*v2(3)
rotall(3,1,ie)=v3(1)**2
rotall(3,2,ie)=v3(2)**2
rotall(3,3,ie)=v3(3)**2
rotall(3,4,ie)=v3(1)*v3(2)
rotall(3,5,ie)=v3(2)*v3(3)
rotall(3,6,ie)=v3(1)*v3(3)
rotall(4,1,ie)=2.0*v1(1)*v2(1)
rotall(4,2,ie)=2.0*v1(2)*v2(2)
rotall(4,3,ie)=2.0*v1(3)*v2(3)
rotall(4,4,ie)=v1(1)*v2(2)+v2(1)*v1(2)
rotall(4,5,ie)=v1(2)*v2(3)+v2(2)*v1(3)
rotall(4,6,ie)=v1(3)*v2(1)+v2(3)*v1(1)
rotall(5,1,ie)=2.0*v2(1)*v3(1)
rotall(5,2,ie)=2.0*v2(2)*v3(2)
rotall(5,3,ie)=2.0*v2(3)*v3(3)
rotall(5,4,ie)=v2(1)*v3(2)+v3(1)*v2(2)
rotall(5,5,ie)=v2(2)*v3(3)+v3(2)*v2(3)
rotall(5,6,ie)=v2(3)*v3(1)+v3(3)*v2(1)
rotall(6,1,ie)=2.0*v3(1)*v1(1)
rotall(6,2,ie)=2.0*v3(2)*v1(2)
rotall(6,3,ie)=2.0*v3(3)*v1(3)
rotall(6,4,ie)=v3(1)*v1(2)+v1(1)*v3(2)
rotall(6,5,ie)=v3(2)*v1(3)+v1(2)*v3(3)
rotall(6,6,ie)=v3(3)*v1(1)+v1(3)*v3(1)

c Rename displacement for code compatability (Global Coord)
  call kmcpdp(edispg,1,ie)

c Transform rotations into local coordinates
  do 60 i=1,4
    do 60 j=1,3
      edisp((i-1)*6+j)=edispg((i-1)*6+j)
      edisp((i-1)*6+3+j)=(edispg((i-1)*6+4)*tlinv(j,1) +
1      edispg((i-1)*6+5)*tlinv(j,2) +
2      edispg((i-1)*6+6)*tlinv(j,3))
    60 continue

c Put back displacements
  call kmcpdp(edisp,0,ie)

c *** End Element Setup ***
  70 continue

c *** Begin Integration Loop ***
  do 230 iz=1,nip
    ipt = iz

c Get Gauss Point

```

```

        if(irl.eq.0)then
            zta = zet(iz,nip)
        elseif (irl.gt.0) then
            zta = 2.0 * (iz-1)/(nip-1)-1.0
        else
            zta = rule(iz,1,nrl)
            mtu = nint(rule(iz,3,nrl))
            mtv=mxt(lft)
            if (mtu.ne.0) then
                mxt(lft)=mtu
            endif
        endif

        call kmshap(zta,shapel)

c *** Begin Element Strain Calculation ***
        do 130 ie=lft,llt

c Jacobian, inverse and its determinant (stored)
            hz=thick(ie)*0.5*(shapel(2)*(1.0-zbr)-shapel(1)*(1.0+zbr))
            hzdt=thick(ie)*0.5
            aj(1,1)=0.25*(-x1(ie)+x2(ie)+x3(ie)-x4(ie))
            aj(2,1)=0.25*(-x1(ie)-x2(ie)+x3(ie)+x4(ie))
            aj(3,1)=hzdt*gl13(ie)
            aj(1,2)=0.25*(-y1(ie)+y2(ie)+y3(ie)-y4(ie))
            aj(2,2)=0.25*(-y1(ie)-y2(ie)+y3(ie)+y4(ie))
            aj(3,2)=hzdt*gl23(ie)
            aj(1,3)=0.25*(-z1(ie)+z2(ie)+z3(ie)-z4(ie))
            aj(2,3)=0.25*(-z1(ie)-z2(ie)+z3(ie)+z4(ie))
            aj(3,3)=hzdt*gl33(ie)

            call kminv3(aj,ajinv,det)
            ajdet(ie) = det

c compute global derivatives and strain-nodal displacement matrix
            do 80 in=1,4
                derivg(1,in)=ajinv(1,1)*deriva(in)+ajinv(1,2)*derivb(in)
                derivg(2,in)=ajinv(2,1)*deriva(in)+ajinv(2,2)*derivb(in)
                derivg(3,in)=ajinv(3,1)*deriva(in)+ajinv(3,2)*derivb(in)
                derilg(1,in)=ajinv(1,3)*hzdt
                derilg(2,in)=ajinv(2,3)*hzdt
                derilg(3,in)=ajinv(3,3)*hzdt
            80    continue

c Zero out B matrix
            do 90 i=1,6
                do 90 j=1,24
                    90    bmtx(i,j,ie)=0.0

c Calculate B Matrix
            do 100 in=1,4
                i1=(in-1)*6+1
                i2=i1+1
                i3=i2+1
                i4=i3+1
                i5=i4+1
                i6=i5+1
                gk1=derivg(1,in)*hz+0.25*derilg(1,in)
                gk2=derivg(2,in)*hz+0.25*derilg(2,in)
                gk3=derivg(3,in)*hz+0.25*derilg(3,in)

```

c

```

bmtx(1,i1,ie)=derivg(1,in)
bmtx(1,i4,ie)=gk1*(-gl12(ie))
bmtx(1,i5,ie)=gk1*gl11(ie)
bmtx(1,i6,ie)=gk1*gl13(ie)
bmtx(2,i2,ie)=derivg(2,in)
bmtx(2,i4,ie)=gk2*(-gl22(ie))
bmtx(2,i5,ie)=gk2*gl21(ie)
bmtx(2,i6,ie)=gk2*gl23(ie)
bmtx(3,i3,ie)=derivg(3,in)
bmtx(3,i4,ie)=gk3*(-gl32(ie))
bmtx(3,i5,ie)=gk3*gl31(ie)
bmtx(3,i6,ie)=gk3*gl33(ie)
bmtx(4,i1,ie)=derivg(2,in)
bmtx(4,i2,ie)=derivg(1,in)
bmtx(4,i4,ie)=gk2*(-gl12(ie))+gk1*(-gl22(ie))
bmtx(4,i5,ie)=gk2*gl11(ie)+gk1*gl21(ie)
bmtx(4,i6,ie)=gk2*gl13(ie)+gk1*gl23(ie)
bmtx(5,i2,ie)=derivg(3,in)
bmtx(5,i3,ie)=derivg(2,in)
bmtx(5,i4,ie)=gk3*(-gl22(ie))+gk2*(-gl32(ie))
bmtx(5,i5,ie)=gk3*gl21(ie)+gk2*gl31(ie)
bmtx(5,i6,ie)=gk3*gl23(ie)+gk2*gl33(ie)
bmtx(6,i1,ie)=derivg(3,in)
bmtx(6,i3,ie)=derivg(1,in)
bmtx(6,i4,ie)=gk3*(-gl12(ie))+gk1*(-gl32(ie))
bmtx(6,i5,ie)=gk3*gl11(ie)+gk1*gl31(ie)
bmtx(6,i6,ie)=gk3*gl13(ie)+gk1*gl33(ie)

```

100 continue

c Retrieve rotated displacements
call kmcpdp(edisp,1,ie)

c Calculate Strain
do 110 i=1,6
 estrainp(i)=0.0
 do 110 k=1,24
110 estrainp(i)=estrainp(i)+bmtx(i,k,ie)*edisp(k)

c Transform strain to local coordinates system
do 120 i=1,6
 estrain(i)=0.0
 do 120 k=1,6
120 estrain(i)=estrain(i)+rotall(i,k,ie)*estrainp(k)

c Store Strain increments
d1(ie) = estrain(1)*dt1
d2(ie) = estrain(2)*dt1
d3(ie) = estrain(3)*dt1
d4(ie) = estrain(4)*dt1
d5(ie) = estrain(5)*dt1
d6(ie) = estrain(6)*dt1
zeta(ie)=zta*thick(ie)
einc(ie) = 0.0

c *** End of element strain loop ***
130 continue

c Constitutive Model Evaluation (ie strain to stress)

```

        call kmcon (nmtcon,auxvec,cm,lav,mte,nip,ipt,csprop(1,mxe),
1      dampk,ym,prv,mxe)

c if strain output is selected, store strain tensor for this ipt
  if (istrn.ne.0) then
    if (irl.eq.0) then
      if (iz.eq.isrn(1,nip)) then
        call tbstbo (strain(1,nnml+1), cm, nnml, mte, emain(1,1))
      endif
      if (iz.eq.isrn(2,nip)) then
        call tbstbo (strain(7,nnml+1), cm, nnml, mte, emain(4,1))
      endif
    else
      if (iz.eq.1 ) then
        call tbstbo (strain(1,nnml+1), cm, nnml, mte, emain(1,1))
      endif
      if (iz.eq.nip) then
        call tbstbo (strain(7,nnml+1), cm, nnml, mte, emain(4,1))
      endif
    endif
  endif

c Get gauss weight
  if (irl.eq.0) then
    fac=wgts(iz,nip)
  elseif (irl.gt.0) then
    fac=2./(nip-1)
  else
    fac=rule(iz,2,nrl)
    mxt(lft)=mtv
  endif

c *** Begin Element Force Loop ***
  do 200 ie=lft,llt

c Update epsilon zz, if selected
    if (istupd.ne.0) str33(ie)=str33(ie)+0.5*fac*d3(ie)

c Copy Element stress tensor to local variable
    estress(1)=sig1(ie)
    estress(2)=sig2(ie)
    estress(3)=sig3(ie)
    estress(4)=sig4(ie)
    estress(5)=sig5(ie)
    estress(6)=sig6(ie)

c Get transpose of rotation matrix
    do 140 i=1,6
      do 140 j=1,6
140    rotrn(i,j)=rotall(j,i,ie)

c Rotate stresses to global coordinates
    do 150 i=1,6
      estressg(i)=0.0
      do 150 k=1,6
150    estressg(i)=estressg(i)+rotrn(i,k)*estress(k)

c Get transpose of B matrix
    do 160 i=1,24
      do 160 j=1,6

```

```

160   bmtxt(i,j)=bmtx(j,i,ie)

c Calculate element forces
  do 170 i=1,24
    eforce(i)=0.0
    do 170 k=1,6
170   eforce(i)=eforce(i)+bmtxt(i,k)*estressg(k)

c Rotate moments
  do 180 i=1,4
    do 180 j=1,3
      eforceg((i-1)*6+j)=eforce((i-1)*6+j)
      eforceg((i-1)*6+3+j)=eforce((i-1)*6+4)*tlall(j,1,ie)
1      + eforce((i-1)*6+5)*tlall(j,2,ie)
2      + eforce((i-1)*6+6)*tlall(j,3,ie)
180   continue

c Sum up forces for each node
  detwt=ajdet(ie)*4.0*fac
c      4.0 comes from x and y gauss weights (2.0 each)
  do 190 i=1,4
    sforce1(ie,i) = sforce1(ie,i) + eforceg((i-1)*6+1)*detwt
    sforce2(ie,i) = sforce2(ie,i) + eforceg((i-1)*6+2)*detwt
    sforce3(ie,i) = sforce3(ie,i) + eforceg((i-1)*6+3)*detwt
    sforce4(ie,i) = sforce4(ie,i) + eforceg((i-1)*6+4)*detwt
    sforce5(ie,i) = sforce5(ie,i) + eforceg((i-1)*6+5)*detwt
    sforce6(ie,i) = sforce6(ie,i) + eforceg((i-1)*6+6)*detwt
190   continue

c *** End of Element Force Loop ***
200   continue

c Do for compatability with output routines; b(ipi) = volf, b(iph) = epf
  fac1=1./fac
  do 220 i=lft,llt
    fga(i)=fac*fga(i)
    fgb(i)=fac*fgb(i)
    if (ipi.eq.iph) go to 210
    b(ipi+i+nnml)=b(ipi+i+nnml)+fga(i)
    b(iph+i+nnml)=b(iph+i+nnml)+ep(i)*fga(i)
210   xies(i) =xies(i)+.5*fga(i)*sarea(i)*einc(i)
    fga(i) =faci*fga(i)
    fgb(i) =faci*fgb(i)
220   continue

*** End of Integration Loop
230   continue

c Change shell thickness if required
c *** Must be an error in indexing - causes "Access Violation"
  if (istupd.ne.0) then
    do 240 ie=lft,llt
      indfib = nnml+1+ie
      str33(ie)=1.+str33(ie)
      fibl(1,indfib)=str33(i)*fibl(1,indfib)
      b(istupd+ix1(ie))= max(b(istupd+ix1(ie)),fibl(1,indfib))
      b(istupd+ix2(ie))= max(b(istupd+ix2(ie)),fibl(1,indfib))
      b(istupd+ix3(ie))= max(b(istupd+ix3(ie)),fibl(1,indfib))
      b(istupd+ix4(ie))= max(b(istupd+ix4(ie)),fibl(1,indfib))
240   continue

```

```

endif

c Compute hourglass forces and add to internal forces, then
c store in appropriate global variables
sndspd=sqrt(sndspd*rho)

call kmfrc(rhs,rhs(1+neq),fibl(2,nm1+1),mte,ibls,sforce1,
1      sforce2,sforce3,sforce4,sforce5,sforce6,ndlist,nlstm,
2      ym,rotall,b(lc11),b(lc13))

return
end

```



```

        subroutine sets44(cm)
        implicit double precision (a-h,o-z)
        dp
c *****
c called by matin to set parameters for material type 44
c
c   call sets44 (cm(1,n))
c
c   ***** McDermott 1999 *****
c   dimension cm(48),strain(16),stress(16),slope(16)

c   Number of segments in stress-strain curve
        iseg=int(cm(11))
        estrain = cm(4) / cm(1)
c   Copy endpoints to temp array
        do 10 i=1,16
            strain(i)=cm(i+11)
            stress(i)=cm(i+27)
10 continue

c   Calculate slopes
        if (iseg.ge.1) then
            slope(1) = (stress(1)-cm(4))/(strain(1)-estrain)
            if (iseg.ge.2) then
                do 20 i=2,iseg
20             slope(i) = (stress(i)-stress(i-1))/(strain(i)-strain(i-1))
            endif
        endif

c   Redo cm array (zero unused values)
        do 30 i=1,16
            if (iseg.ge.i) then
                cm(2*(i-1)+12) = slope(i)
                cm(2*(i-1)+13) = strain(i)
            else
                cm(2*(i-1)+12) = 0.0
                cm(2*(i-1)+13) = 0.0
            endif
30 continue
        return
        end

```

```

subroutine shl44s (cm, capa)
implicit double precision (a-h,o-z)
include 'nlqpar.inc'
c *****
c Elastic-plastic isotropic material with void growth and nucleation
c and piecewise-linear stress-strain curve
c McDermott 1999
c *****
common/bk02/iburn, isdo, dt1, dt2
common/aux2/d1(nlq), d2(nlq), d3(nlq), d4(nlq), d5(nlq), d6(nlq),
1 wzzdt(nlq), wyydt(nlq), wxxdt(nlq), einc(nlq)
common/aux14/
1 sig1(nlq), sig2(nlq), sig3(nlq), sig4(nlq),
2 sig5(nlq), sig6(nlq), ep(nlq), ep1(nlq), ep2(nlq),
3 ep4(nlq), ep5(nlq), ep6(nlq), ep3(nlq), effs(nlq),
4 phi(nlq), sig0(nlq), aux(nlq, 55)
common/aux18/dd(nlq), def(nlq)
common/aux33/
1 ix1(nlq), ix2(nlq), ix3(nlq), ix4(nlq), ix5(nlq, 4), mxt(nlq)
common/aux35/rhoa(nlq), cb(nlq), davg(nlq), p(nlq)
common/aux36/lft, llt
common/sound/sndspd, sndsp(nlq), diagm(nlq), sarea(nlq), dx1(nlq)
common/hourg/ymod, gmod, ifsv
common/hour11/ebars(nlq), ebarmn(nlq), eyld(nlq), etanmd(nlq)
common/failul/sieu(nlq), failu(nlq)
common/ssbsis/h(8, 5, 5), hpr(8, 5, 5), hps(8, 5, 5), hpt(8, 5, 5), ipt,
1 nip, wgts(5, 5), zet(5, 5)
dimension fail(nlq), cm(*), strain(20), slope(20)
data third/-.3333333333333333/
data sfac/0.8333333333333333/
data eps/1.0e-4/
c
c Retrieve Material Properties from CM array
mx=48*(mxt(lft)-1)
ym=cm(mx+1)
pr=cm(mx+2)
phi0=cm(mx+3)
syp=cm(mx+4)
q1=cm(mx+5)
q2=cm(mx+6)
q3=cm(mx+7)
fn=cm(mx+8)
en=cm(mx+9)
sn=cm(mx+10)
iseg=int(cm(mx+11))
ystrain=syp/ym
strain(1)=ystrain
if(iseg.gt.0)then
do 10 i=1, iseg
slope(i)=cm(mx+10+i*2)
strain(i+1)=cm(mx+11+i*2)
10 continue
endif
ymod=ym
gmod=ym/(2.0*(1.0+pr))
bulk=ym/(3.0*(1.0-2.0*pr))
sndspd=ym/(1.0-pr**2)
c Set up Element Values

```

```

do 60 i=lft,llt
  einc(i)=d1(i)*sig1(i)+d2(i)*sig2(i)+d4(i)*sig4(i)+d5(i)*sig5(i)
1   +d6(i)*sig6(i)+d3(i)*sig3(i)
  plastrn = ep(i)
  phit = phi(i)
  stress0 = sig0(i)
  sg1=sig1(i)+sndspd*(d1(i)+pr*d2(i))
  sg2=sig2(i)+sndspd*(pr*d1(i)+d2(i))
  sg3=sig3(i)+ym*d3(i)
  sg4=sig4(i)+gmod*d4(i)
  sg5=sig5(i)+sfac*gmod*d5(i)
  sg6=sig6(i)+sfac*gmod*d6(i)
  press=third*(sg1+sg2+sg3)
  s1=sg1+press
  s2=sg2+press
  s3=sg3+press
  q=sqrt(1.5*(s1**2+s2**2+s3**2+2.0*sg4**2+
1   2.0*sg5**2+2.0*sg6**2))
  F=(q/stress0)**2+2.0*q1*phit*
1   cosh(-1.5*q2*press/stress0)-(1.0+q3*phit**2)
  if (F.gt.0.0) then
    iter = 0
    dep = 0.0
    deq = 0.0

c Begin Iteration Loop
20   c1=3.0*q2/(2.0*stress0**2)
      Fq1=2.0*q/(stress0**2)
      Fq2=2.0/(stress0**2)
      Fp1=-2.0*q1*phit*c1*sinh(-c1*press)
      Fp2=2.0*q1*phit*c1**2*cosh(-c1*press)
      Fsl=-q**2/(2.0*stress0**3)+2.0*q1*phit*c1*press*sinh(-c1*press)
      Fsp=2.0*q1*phit*(-c1**2*press*cosh(-c1*press)+c1/stress0*
1     sinh(-c1*press))
      Fsq=-q/stress0**3

c Get Partial of Yield Stress with respect to dep
      stress1 = syp
      es1 = stress0/ym
      e1 = -press*dep/((1-phit)*stress0)
      ps1 = plastrn + e1 + es1
      iflag = 0
      do 30 j=1,iseq
        if(iflag .eq. 0) then
          if(ps1 .lt. strain(j+1))then
            stress1 = stress1 + slope(j)*(ps1-strain(j))
            iseg1 = j
            iflag = 1
          else
            stress1 = stress1 + slope(j)*(strain(j+1)-strain(j))
          endif
        endif
      30 continue
c Numerical Error Trap
      if((stress1.eq.stress0) .or. (abs(e1) .eq. 0.0)) then
        dsdep = slope(iseg1)
      else
        dsdep = (stress1-stress0)/e1
      endif
      if (abs(dsdep) .gt. ym) dsdep = slope(iseg1)

```

```

c Get Partial of Yield Stress with respect to deq
  stress1 = syp
  e1 = q*deq/((1-phit)*stress0)
  psl = plastrn + e1
  iflag = 0
  do 40 j=1, iseg
    if(iflag .eq. 0) then
      if(psl .lt. strain(j+1)) then
        stress1 = stress1 + slope(j)*(psl-strain(j))
        iseg1 = j
        iflag = 1
      else
        stress1 = stress1 + slope(j)*(strain(j+1)-strain(j))
      endif
    endif
  40 continue
c Numerical Error Trap
if((stress1.eq.stress0) .or. (abs(e1) .eq. 0.0)) then
  dsdeq = slope(iseg1)
else
  dsdeq = (stress1-stress0)/e1
endif
if (abs(dsdeq) .gt. ym) dsdeq = slope(iseg1)

c Apply Newtons Method
a22 = bulk*Fp1+Fsl*dsdeq
a12 = 3.0*gmod*Fq1+Fsl*dsdeq
a21 = -(Fq1+deq*(bulk*Fp2+Fsp*dsdeq)+dep*Fsq*dsdeq)
a11 = Fp1+dep*(-3.0*gmod*Fq2+Fsq*dsdeq)+deq*Fsp*dsdeq
deta = a11*a22-a12*a21
a11 = a11/deta
a12 = a12/deta
a21 = a21/deta
a22 = a22/deta
b1 = -1.0*F
b2 = -1.0*dep*Fq1-deq*Fp1
dep = dep + a11*b1 + a12*b2
deq = deq + a21*b1 + a22*b2

c Reset Key Parameters
phit = phi(i)
plastrn = ep(i)
stress0 = sig0(i)

c Plastic Strain
deplst1=dep/3.0+1.5*deq*s1/q
deplst2=dep/3.0+1.5*deq*s2/q
deplst3=dep/3.0+deq*sg3/q
deplst4=1.5*deq*sg4/q
deplst5=1.5*deq*sg5/q
deplst6=1.5*deq*sg6/q
degrowth = deplst1+deplst2+deplst3

c Adjust stress, q, pressure, etc.
sg1 = sig1(i) + sndspd*((d1(i)-deplst1)+pr*(d2(i)-deplst2))
sg2 = sig2(i) + sndspd*(pr*(d1(i)-deplst1)+(d2(i)-deplst2))
sg3 = sig3(i) + ym*(d3(i)-deplst3)
sg4 = sig4(i) + gmod*(d4(i)-deplst4)
sg5 = sig5(i) + sfac*gmod*(d5(i)-deplst5)

```

```

    sg6 = sig6(i) + sfac*gmod*(d6(i)-deplst6)
    press = third*(sg1+sg2+sg3)
    s1 = sg1 + press
    s2 = sg2 + press
    s3 = sg3 + press

c Calculate Increase in Plastic Strain (must be > 0)
    deltep = (q*deq-press*dep)/((1-phit)*stress0)
    if(deltep .lt. 0.0) deltep = 0.0
    plastrn = plastrn + deltep

c Compute Void Nucleation and Growth
    if(fn .ne. 0.0) then
        anucl = fn/(sn*2.50662827463)*exp(-0.5*((plastrn-en)/sn)**2)
    else
        anucl = 0.0
    endif
    phil = phit+(1-phit)*degrowth+anucl*deltep
c Decrease in porosity not allowed
    if(phil.gt.phit) phit = phil
c Turn off void effects if q1 is set to zero (elastic-plastic only)
    if(q1 .eq. 0.0) phit = 0.0

c Calculate Strain Hardening
    iflag = 0
    ps1 = plastrn + stress0/ym
    stress0 = syp
    do 50 j=1, iseg
        if(iflag.eq.0) then
            if(ps1.lt.strain(j+1)) then
                estrain = -stress0/ym
                stress0 = stress0 + slope(j)*(ps1-strain(j))
                estrain = estrain + stress0/ym
                stress0 = stress0+slope(j)*estrain
                iflag = 1
            else
                stress0 = stress0 + slope(j)*(strain(j+1)-strain(j))
                ps1 = plastrn + stress0/ym
            endif
        endif
    50    continue

c Calculate new Yield Function
    q=sqrt(1.5*(s1**2+s2**2+s3**2+2.0*sg4**2+
1      2.0*sg5**2+2.0*sg6**2))

    F=(q/stress0)**2+2.0*q1*phit*
1      cosh(-1.5*q2*press/stress0)-(1.0+q3*phit**2)
    iter = iter+1

c Check for failure to converge
    if(iter.gt.500) then
        write(*,*) 'SHL44S - Failed to Converge'
        call adios(2)
    endif
    if(abs(F).gt.eps) goto 20
    else
c No increase in plastic strain, set tensor to zero
        deplst1=0.0
        deplst2=0.0

```

```

        deplst3=0.0
        deplst4=0.0
        deplst5=0.0
        deplst6=0.0
    endif
c DONE! Load up all required variables
    sig0(i) = stress0
    p(i) = press
    sig1(i) = sg1
    sig2(i) = sg2
    sig3(i) = sg3
    sig4(i) = sg4
    sig5(i) = sg5
    sig6(i) = sg6
    ep(i) = plastrn
    ep1(i) = ep1(i) + deplst1
    ep2(i) = ep2(i) + deplst2
    ep3(i) = ep3(i) + deplst3
    ep4(i) = ep4(i) + deplst4
    ep5(i) = ep5(i) + deplst5
    ep6(i) = ep6(i) + deplst6
    phi(i) = phit
    ebar(i) = ym
    cb(i)=sndspd
    davg(i)=third*(d1(i)+d2(i)+d3(i))
    effs(i)=q
    einc(i)=d1(i)*sig1(i)+d2(i)*sig2(i)+d4(i)*sig4(i)+d5(i)*sig5(i)
1      +d6(i)*sig6(i)+einc(i)+d3(i)*sig3(i)
    eyld(i)=stress0
60 continue
    return
end

```


APPENDIX C. DYSMAS MODIFIED SUBROUTINE LISTINGS.

```
blockdata blkdat
implicit double precision (a-h,o-z)
dp
include 'nlqpar.inc'
c
c ***** Added Number of material constants for material type 44 -
McDermott '99
c
.
.
.
data nconst/0,17,5,3,0,6,12,3,3,4,4,0,1,2,5,6,10,5,2,0,20,14,
1 2,0,2,0,12,5,5,9,12,18,25,0,20,44,11,8,10,25,14,5,26,16,16*0/
.
.
.
end
```

```

subroutine dynai (inelflg)
implicit double precision (a-h,o-z)
.
.
.
C
C **** Added Support for Kwon-McDermott Shell Element Formulation
C (Only change is in format statement 227) - McDermott '99
.
.
.
C *** Changed 227 to add Kwon-McDermott Formulation - McD
227 format(
$ 4x,'shell formulation basis.....',i7/
$10x,'eq.1: hughes-liu shell theory',i7/
$10x,'eq.2: belytschko-lin-tsay shell theory',i7/
$10x,'eq.3: bciz',i7/
$10x,'eq.4: c0-triangular element',i7/
$10x,'eq.5: membrane element',i7/
$10x,'eq.6: yase',i7/
$10x,'eq.7: QPHM',i7/
$10x,'eq.8: kwon-mcdermott shell',i7/
$ 4x,'number of non-reflecting boundary segments.....',i7//
$ 4x,'number of single point constraint nodes.....',i7//
$ 4x,'number of spc coordinate system definitions.....',i7//
$ 4x,'reduction factor for tsmn.....',e10.2//
$ 4x,'# of user specified beam integration rules.....',i7//
$ 4x,'max number of integration points reqd (beams) .',i7//
$ 4x,'# of user specified shell integration rules.....',i7//
$ 4x,'max number of integration points reqd (shells).',i7//
$ 4x,'convergence check interval (dynamic relaxation)',i7//
$ 4x,'convergence tolerance for dynamic relaxation... ',e10.2/)
.
.
.
end

```

```

      subroutine elem2d(ixp,x,rhs,vt,vr,strain,yhatn,fibl,auxvec,
1mtype,ro,cm,csp,prop,nsubgv,mtnum,nfegp,ihgq,hgq,xies,ener,rule,
2mpusr,ishl,tfail,isf,lochvs,qextra,nshel,nncs,ibls,
3 dampk,ym,pr,fails)
      implicit double precision (a-h,o-z)
      dp
      include 'nlqpar.inc'

c
c      main subroutine for calling two-dimensional elements
c
c
c *** Added Kwon-McDermott Shell Formulation - McDermott 1999
c
c
c
c *****
c Kwon-McDermott Shell Formulation
      elseif (iop.eq.8) then
          call kwnmcd(rule,ixp,x,rhs,vt,vr,strain,yhatn,fibl,auxvec,
1              mtype,ro,cm,csp,prop,nsubgv,mtnum,nfegp,ihgq,hgq,xies,ener,
2              mpusr,lav,nmel,nnml,mxe,ibls,dampk(mxe),ym(mxe),pr(mxe),
3              a(ns02+6*nnml))

      endif
c *****
c
c
c
c end

```

```

      subroutine fem3d(mtype,ro,cm,ic,bcs,npc,pld,nod,idirn,ncur,clfac,
1llc,nvel,vx,vy,vz,fval,rd,ilcw,nswn,numtp,nodtie,tim,iparm,irects,
2irectm,nsv,msr,nsegs,nsegm,lsv,lmsr,ilocs,ilocm,stfs,stfm,
3irtls,irtlm,xmsm,e,crst,b,tcode,u,v,a,x,xms,ac,nsbgv,mtnum,
4nfegp,auxvec,rhsi,zfac,ieost,eosp,ihgg,hgg,iqtype,bkqs,rbu,rbv,
5rba,rbi,rbm,rbcor,nrbn,nrba,nrb,xrb,yrb,zrb,axrb,ayrb,azrb,
6rbfx,rbfy,rbfz,rbcods,mxrb,xyzkcn,lpntbk,lbcket,chrlen,ethik,
7fric,iseq,fdat,fthik,icls,irctsi,irctmi,ilcf,itcode,atcode,ifo,
8slvfrc,msrfrc,ener,rots,failz,sfail,fl9s,tfail,isf,drdsps,nnfpln,
9thkslv,thkmsr,failh,fails,ihsnd,islnd,ifalhl,ifalsl,ihl2sg,
1isl2sg,isg2el,iacthl,iactsl,irecta,stfa,thka,lsndsc,stfsnd,
2thksnd,thicks,ishltp,islfm,xnew,ftemp,xls,xlm,cntrls,vract,mlbf,
3bfact,rectd,xndchr,thkseg,itytp,iclsl,lclsa,nclsa,acclsl,accmsr,
cfu      4cornew,tmad,madmat,dbldat)
      4cornew,tmad,madmat,dbldat,pfric,weldf)
C*****
****
C
C *** Fixed bug in disp,vel,accel,etc. plot file write counter -
McDermott 1999
C
.
.
.
if(ncycle.ge.1)goto 10
cte
      output=.false.
      ncredit=0
      ioatb=0
      nfedit=0
C
C.... check for variable plot interval, get load curve number
C
      ipltlc=0
      if( pltc .lt. 0. ) then
        ipltlc=abs(pltc)
      endif
C *** Changed from pltout=pltc to pltout=0. to print 1st time step - McD
'99
.
.
.
C *****
C   BUG FIX FOR PRINTOUT INTERVAL - MCDERMOTT
C   There appears to be code already written to do this, but some
C   are commented out. This was a temporary fix I implemented.
C
      if (tt.lt.prtout) go to 160
      prtout=prtout+prtc
      if (prtc.gt.endtim) go to 160
C *****
.
.
.
end

```

```

      subroutine in3dis(x,matype,den,prop,csprop,ipss,yhat,fibl,
1 auxvec,icnt,xnrvect,xmst,xmsr,numels,mx,ix,rbm,strain,beta,
2 fval,tnew,ishlfn,nshl,fails)
      implicit double precision (a-h,o-z)
      dp
C
C called by ibmsh for shell initialization
C
C shell initialization (compute mass at nodes)
C
C      call in3dis(a(lc11),a(n1),a(n2),a(n3),a(n4f),a(lc1s),
C      1 a(ns03),a(ns05),a(ns06),a(lc10),a(lc10+numnp),a(lc14),
C      2 a(lc14+numnp),numels,ix,ix(2),a(n53),a(ns01),a(ns04),
C      3 a(n19),a(ntmpl),a(n1+nmmat),a(nsl7),a(nsl7+numels))
C
C      a shell element
C
C *** Added initialization for material type 44 - McDermott 1999
C
C
C
C *** Kwon-McDermott - Initialize AUXVEC for material 44
C
      if (mte.eq.44) then
        do 17 i=1,nmtcon
17      auxvec(i+lav)=0.0
C      Set initial porosity and yield stress
        do 19 i=1,nip
          auxvec((i-1)*npi+15+lav)=prop(3,mx)
          auxvec((i-1)*npi+16+lav)=prop(4,mx)
19      continue
        endif
C
C *****
C
C
C
C
end

```

```

      subroutine matin(ieost,eosp,ihgq,hgq,iqtype,bqs,csprop,
1 mtype,ishlfm,ro,cm,idfrs,nmmat,lc2,mmauxs,iortho,ifb,isf,
2 numelh,numels)
      implicit double precision (a-h,o-z)
      dp
c
c called by dynai to read in material data
c
c      call matin (a(n4a),a(n4b),a(n4c),a(n4c+nmmat),a(n4d),a(n4e),
c      1 a(n4f),a(n1),a(n1+nmmat),a(n2),a(n3),idfrs,nmmat,lc2,mmauxs,
c      2 iortho,ifb,isf,numelh,numels)
c
c *** Added Material #44 (Elasto-plastic w/void effects) and allow
c      Kwon-McDermott shell formulation to have a non-zero reference
c                                     McDermott 1999
c
c
c
c
c if material type is 41
c.... fabric model
c
      if (mtype(n).eq.41) then
        ifst=1
        ilst=3
        do 82 j=1,5
          call gttxsq (txts,lcount)
          if (j.eq.1) then
            read (unit=txts,fmt=220,err=200) (cm(i,n),i=ifst,ilst),cm(48,n),
1 (cm(i,n),i=45,47)
          elseif (j.eq.2) then
            read (unit=txts,fmt=220,err=200) (cm(i,n),i=ifst,ilst),
1 cm(42,n),cm(43,n),cm(44,n),cm(41,n)
          elseif (j.gt.2.and.j.lt.5) then
            read (unit=txts,fmt=220,err=200) (cm(i,n),i=ifst,ilst)
          elseif (j.eq.5) then
            read (unit=txts,fmt=220,err=200) (cm(i,n),i=ifst,ilst)
          endif
          ifst=ifst+3
          ilst=ilst+3
62 continue
          call gttxsq (txts,lcount)
          read (unit=txts,fmt=220,err=200) (cm(i,n),i=16,18),(cm(i,n),i=26,3
1 0)
          endif
c
c *****
c Kwon-McDermott
c material type 44 (Elastic Plastic with Void Effects)
      if (mtype(n).eq.44) then
        call gttxsq (txts,lcount)
        read (unit=txts,fmt=220,err=200) (cm(i,n),i=1,4)
        call gttxsq (txts,lcount)
        read (unit=txts,fmt=220,err=200) (cm(i,n),i=5,11)
        iseg=cm(11,n)
        ifst=12
        do 25 j=1,4
          if (j.eq.3) iseg=cm(11,n)
          call gttxsq (txts,lcount)

```

```

    if (iseg.ge.8) then
      lread = 8
      iseg = iseg - 8
    else
      lread = iseg
      iseg = 0
    endif
    if (lread.gt.0) then
      read (unit=txts,fmt=220,err=200) (cm(i,n),i=ifst,ifst+lread)
    endif
    ifst = ifst + 8
25  continue
    call sets44(cm(1,n))
    call blkcpy(cm(1,n),prop,48)
  endif
c *****
.
.
.
c
c if element type not shell goto 100
c
  90 if (itype.ne.2) go to 100
    csprop(14,n)=0.0
    csprop(15,n)=0.0
    csprop(16,n)=0.0
c *** Changed to allow Kwon-McDermott shell to have non-zero ref - McD
'99
    if ((ishlfm(n).eq.1.and.irnxx.ge.0).or.(ishlfm(n).eq.8)) then
.
.
.
290 format(' ***warning*** the reference surface must be the.',/,
1  'midsurface for all shell elements except hughes-liu or',/,
2  'kwon-mcdermott.',/)
.
.
.
end

```

```

      subroutine nbsint (irect,bulk,shear,nrt,cm,matype,eosp,ieost,
1  numelh,bh,jx1,jx7,x,ro,nmmat)
      implicit double precision (a-h,o-z)
      dp
c
c  called by initlz
c
c  if number of non-reflecting boundary segments (nnrbs) greater
c  than zero call subroutine nbsint to initialize transmitting
c  boundary segments
c
c      if (nnrbs.gt.0) then
c          call nbsint (b(lrb1),b(lrb2),b(lrb3),nnrbs,cm,matype,eosp,ieost,
c          & numelh,ipsh,b(ibins1),b(ibins2),x,ro,nmmat)
c      endif
c
c  *** Added support for Mat #44 - McDermott 1999
c
c
c
c  if (mt.eq.43) then
c      bkm(mx)=cm(48*(mx-1)+1)/3.
c      shm(mx)=cm(48*(mx-1)+1)/2.
c  endif
c  *** Elasto-Plastic with Void Effects
c      if (mt.eq.44) then
c          bkm(mx)=cm(48*(mx-1)+1)/(3.*(1.-2.*cm(48*(mx-1)+2)))
c          shm(mx)=cm(48*(mx-1)+1)/(2.*(1.+cm(48*(mx-1)+2)))
c      endif
c  ***** McD '99
c
c
c
end

```



```

      subroutine penstf (x,nrb,xyzkcn,rbncod,ipsh,ipss,ipsb,cm,matype,
1  eosp,ieost,numelh,numels,numelb,nmmat,ro,zf,thicks,fibers,
2  ishlfm)
      implicit double precision (a-h,o-z)
      dp
c
c  called by initlz to compute locations of extra
c  rigid body points
c
c      call penstf (x,nrb,xyzkcn,rbncod,ipsh,b(lc1s),b(lc1b),cm,matype,
c      leosp,ieost,numelh,numels,numelb,nmmat,ro,zfcs,b(ns05),
c      2 b(nb05),b(n1+nmmat))
c
c
c
c      if (mt.eq.41) bkm(mx)=cm(mx48m1+21)
c *** Elasto-Plastic with Void Effects - McD '99
c      if (mt.eq.44) bkm(mx)=cm(mx48m1+1)/(3.*(1.-2.*cm(mx48m1+2)))
c *****
c
c
c
end

```

```

      subroutine printm (n,mod,ro,cm,ieost,eosp,ihgq,hgq,iqtype,bqs,
1 csprop,head,itYPE,angles,thrmpr,nip,ishlfm)
      implicit double precision (a-h,o-z)
      dp
c
c
c called by matin to write out (echo) material properties
c
c 170 call printm (n,mtype(n),ro(n),cm(1,n),ieost(n),eosp(1,n),ihgq(n)
c   1 ,hgq(n),iqtype(n),bqs(1,n),csprop(1,n),head,itYPE,a(langle),a
c   2 (lthmpr),nip,ishlfm(n))
c
c      subroutine to print out material properties
c
c *** Added Mate #44 (Elasto-Plastic with Void Effects) - McDermott 1999
c
c
c
c      go to (10,20,30,40,50,60,70,80,90,100,110,120,130,140,150,160,170,
1 180,190,10,200,210,230,250,260,270,272,30,274,30,
1 275,276,277,278,279,281,282,283,284,105,107,285,286,287), mod
c
c
c
c      model - 44 Elastic-Plastic w/Void Growth & Nucleation, piecewise
c              linear strain hardening
c                      McDermott 1999
c
287 write (13,1700) (cm(i),i=1,10),int(cm(11))
      write (13,1701)
      do 288 id=1,int(cm(11))
          ind1 = 13+2*(id-1)
          ind2 = 12+2*(id-1)
          write (13,1702) (cm(ind1),cm(ind2))
288 continue
      write (13,1703)
      go to 280
c
c
c
321 format(
$ '      eq.30 closed form update elastic-plastic for shells '//,
$ '      eq.31 frazer-nash hyperelastic rubber'//,
$ '      eq.32 ramberg osgood elastic-plastic'//,
$ '      eq.33 hill general anisotropic plasticity'//,
$ '      eq.34 hill normal anisotropic plasticity for shells'//,
$ '      eq.35 elastic-plastic with forming limit diagram'//,
$ '      eq.36 brittle damage (experimental) '//,
$ '      eq.37 3-invariant viscoplastic cap '//,
$ '      eq.38 bammann plasticity '//,
$ '      eq.39 sandia damage '//,
$ '      eq.40 fahrenheit brittle damage '//,
$ '      eq.41 fabric '//,
$ '      eq.42 MTS '//,
$ '      eq.43 Low Density Polyurethane Foam '//,
$ '      eq.44 Elastic-Plastic with Void Effects'//)

```

```

810 format(
$ 5x,'shell formulation ..... =' ,i5/
$ 5x,'    eq. 1: hughes-liu          '/'
$ 5x,'    eq. 2: belytschko-tsay     '/'
$ 5x,'    eq. 3: bciz                 '/'
$ 5x,'    eq. 4: c0-triangular element '/'
$ 5x,'    eq. 5: membrane element    '/'
$ 5x,'    eq. 6: yase                 '/'
$ 5x,'    eq. 7: QPHM                 '/'
$ 5x,'    eq. 8: kwon-mcdermott element '/'
1 5x,'fiber lengths:                node 1 ..... =' , e12.4/
2 5x,'                                node 2 ..... =' , e12.4/
3 5x,'                                node 3 ..... =' , e12.4/
4 5x,'                                node 4 ..... =' , e12.4//
5 5x,'reference surface:            node 1 ..... =' , e12.4/
6 5x,'    eq. 1.0:top                node 2 ..... =' , e12.4/
7 5x,'    eq. 0.0:middle             node 3 ..... =' , e12.4/
8 5x,'    eq.-1.0:bottom             node 4 ..... =' , e12.4//)

1700 format(
1 5x,'youngs modulus ..... =' , e12.4/
2 5x,'poissons ratio ..... =' , e12.4/
3 5x,'initial porosity ..... =' , e12.4/
4 5x,'yield stress ..... =' , e12.4/
5 5x,'q1 of gursons model ..... =' , e12.4/
6 5x,'q2 of gursons model ..... =' , e12.4/
7 5x,'q3 of gursons model ..... =' , e12.4/
8 5x,'void nucleation content (fn) ..... =' , e12.4/
9 5x,'mean nucleation strain (en) ..... =' , e12.4/
1 5x,'nucleation standard deviation (sn) =' , e12.4/
2 5x,'number of segments in hardening .. =' , i5/)
1701 format(
1 5x,'stress strain curve points:',/
2 7x,'strain          slope')
1702 format(5x,e12.4,'    ',e12.4)
1703 format(/)
c
end

```



```

      SUBROUTINE SCA_ASC ( X, V, ACC, X0, NCPOUT, A, PFRIC)
C
C   CALLED BY PRDAT TO WRITE SCALAR PLOT FILES (ASCII-FORMAT)
C   REIHENFOLGE WICHTIG !!!
C
C               1.) VOLUMENELEMENTE
C               2.) BALKENELEMENTE
C               3.) SCHALENELEMENTE
C               4.) DICKE SCHALENELEMENTE
C
C.. CALL SCA_ASC (A(LC11),A(LC9),A(LC10),A(LC13),A(NCPLL),A(1),A(LC12))
C
c Added Code for AUX14 variables #15 and 16 (Porosity and Yield stress
c   in Material #44) - McDermott - '99
C*****
.
.
.
c **** Used for SCA_GET - McDermott
      dimension val(nlq)
C
.
.
.
C
C   S H E L L   E L E M E N T S
C
      140 IF (NUMELS.EQ.0) GOTO 240
          NELG=NUMELS/nlq
          IF (nlq*NELG.LT.NUMELS) NELG=NELG+1
C
          NEL=0
c Seperate Counter for SCA_GET - McDermott
          nelget=0
          NUMEL = NUMELH + NUMELB + NUMELS
          DO 200 NN=1,NELG
              NMEL=nlq
              IF (NN.EQ.NELG) NMEL=NUMELS-nlq*(NELG-1)
              LNS=49
              CALL SCALARS(A(N1),A(N4F),A(LC11),A(N4A),A(LC1S),A(NS05),
F                  A(NS06),A(NS03),A(NS01),SIG,LNS,A(NS07),A(N4H),
F                  MPUSR,A(N1+NMAT),A(NSTSL),A(NS13),A(NS14),NEL,
F                  NMEL,A(NS02),EMAIN)
C
c ***** Changed to support Void Material (#44) - McDermott
          GOTO (141,142,180,180,180,180,180,180,180,180,180,180,180,
c      F      180,180,180,180,180,180,180,180,180,180,180,180,180,
F      155,156,157,158,180,180,180,180,180,180,165,180,180,180,
F      170,171,172,173,174,175,176,177,178,179) K
.
.
.
c ***** Support for Void Material (#44) - McDermott
c   Void Content
      157 continue
          ival=15
          call sca_get (a(n1),a(n4f),a(lc11),a(n4a),a(lc1s),a(ns06),
      1      sig,lns,a(ns13),a(ns14),nelget,nmel,ival,val,iint)
          do i=1,nmel
              il = i + (nn-1)*nlq

```

```

        write(91, '(1x,i7,1x,E12.5)') i1,sngl(val(i))
    end do
c
    goto 200
c
c  Yield Stress
158  continue
    ival=16
    call sca_get (a(n1),a(n4f),a(lc11),a(n4a),a(lc1s),a(ns06),
1      sig,lms,a(ns13),a(ns14),nelget,nmel,ival,val,iint)
    do i=1,nmel
        i1 = i + (nn-1)*nlq
        write(91, '(1x,i7,1x,E12.5)') i1,sngl(val(i))
    end do
c
    goto 200
c *****
.
.
.
END

```

```

      SUBROUTINE SCA_DYS ( X, V, ACC, X0, NCPOUT, A, PFRIC)
C
C.. CALLED BY PRDAT TO WRITE SCALAR PLOT FILES (DYSMAS/P FORMAT)
C
C.. CALL SCA_DYS (A(LC11),A(LC9),A(LC10),A(LC13),A(NCPLL),A(1),A(LC12))
C
C   Added Support for printing aux14 variables 15 and 16 (porosity and
C   yield stress in Mat #44 - McDermott 1999
C*****
C
C
C   **** Used for SCA_GET - McD '99
C     dimension val(nlq)
C
C
C
C   S H E L L   E L E M E N T S
C
C   140 IF (NUMELS.EQ.0) GOTO 240
C     NELG=NUMELS/nlq
C     IF (nlq*NELG.LT.NUMELS) NELG=NELG+1
C     ikon=ikon+1
C
C     NEL=0
C   c Seperate Counter for SCA_GET - McDermott
C     nelget=0
C     NUMEL = NUMELH + NUMELB + NUMELS
C     NUMEL1 = NUMELH + NUMELB
C     DO 200 NN=1,NELG
C       NMEL=nlq
C       IF (NN.EQ.NELG) NMEL=NUMELS-nlq*(NELG-1)
C       LNS=49
C       CALL SCALARS (A(N1),A(N4F),A(LC11),A(N4A),A(LC1S),A(NS05),
F         A(NS06),A(NS03),A(NS01),SIG,LNS,A(NS07),A(N4H),
F         MPUSR,A(N1+NMAT),A(NSTSL),A(NS13),A(NS14),NEL,
F         NMEL,A(NS02),EMAIN)
C   **** added 57 and 58 - McD
C     GOTO (141,142,180,180,180,180,180,180,180,180,180,180,180,180,
C   F     180,180,180,180,180,180,180,180,180,180,180,180,180,180,
C   F     155,156,157,158,180,180,180,180,180,180,165,180,180,180,180,
C   F     170,171,172,173,174,175,176,177,178,179) K
C
C
C
C   ***** Support for Void Material (#44) - McD '99
C   c Void Content
C   157 continue
C     ival=15
C     call sca_get (a(n1),a(n4f),a(lc11),a(n4a),a(lc1s),a(ns06),
C   1     sig,lns,a(ns13),a(ns14),nelget,nmel,ival,val,iint)
C     do i=1,nmel
C       il = i + (nn-1)*nlq
C       write(91,'(1x,i7,1x,E12.5)') il,sngl(val(i))
C     end do
C
C   goto 200
C
C

```

```

c Yield Stress
158 continue
    ival=16
    call sca_get (a(n1),a(n4f),a(lc11),a(n4a),a(lc1s),a(ns06),
1      sig,lms,a(ns13),a(ns14),nelget,nmel,ival,val,iint)
    do i=1,nmel
        il = i + (nm-1)*nlq
        write(91,'(1x,i7,1x,E12.5)') il,sngl(val(i))
    end do
c
    goto 200
c *****
.
.
.

END

```



```

      SUBROUTINE SCA_TEC ( ACC, NCPOUT, A, PFRIC,
1          nhxpnt,nshpnt,ntxpnt,ixh,ixs,ixt,ss,ival )
C
C CALLED BY PRDAT TO WRITE SCALAR PLOT FILES (TECPLOT-FORMAT)
C REIHENFOLGE WICHTIG !!!
C          1.) VOLUMENELEMENTE
C          2.) BALKENELEMENTE (werden von TECPLOT
nicht          unterstuetzt)
C          3.) SCHALENELEMENTE
C          4.) DICKE SCHALENELEMENTE
C
C CALL SCA_TEC ( A(LC10), A(NCPLL), A(1), A(LC12))
C
C
C Fuchs '97
c *** Variables written to scratch file must be same precision as
c variable used to read back - fixed (removed snl()), added
c support for aux14 variables #15 and 16 (Porosity and Yield stress for
c Material #44 - McDermott '99
C*****
      implicit double precision (a-h,o-z)
      dp
      include 'nlqpar.inc'
      COMMON/BK00/NUMNP,NUMPC,NUMLP,NEQ,NDOF,NLCUR,NUMCL,NUMVC,
1          NDTPTS,NELMD,NMMAT,NUMELH,NUMELB,NUMELS,NUMELT,NUMDP,
2          GRVITY,IDIRGV,NODSPC,NSPCOR
cfu
common/bk03/endtim,prtc,pltc,ndthl,nsthl,nstsl,nstbl,nsttl,mkthf
common/bk03/endtim,prtc,pltc,ngthl,ndthl,nsthl,nstsl,nstbl,
1          nsttl,ncpll,mkthf
COMMON/BK04/PRTOUT,PLTOUT,DT2OLD,SLSFAC,TSSFAC,IHYDRO,
cfu      1          NDTH,NMST,NSTH,NSTS,NSTB,NSTT,IKEDIT
1 ngth,ndth,nmst,nsth,nsts,nstb,nstt,ncpl,ikedit
COMMON/BK05/
1 NH01,NH02,NH03,NH04,NH05,NH06,NH07,NH08,NH09,NH10,
2 NB01,NB02,NB03,NB04,NB05,NB06,NB07,NB08,NB09,NB10,
3 NS01,NS02,NS03,NS04,NS05,NS06,NS07,NS08,NS09,NS10,
4 NT01,NT02,NT03,NT04,NT05,NT06,NT07,NT08,NT09,NT10
REAL*8 HEAD
VAX750
common/bk06/time(2,8),head(12),idmmy,iadd,ifil,maxsiz,ncycle
common/bk07/n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11,n12,n13,n14,n15,
1 n16,n17,n18,n19,n20,n21,n22,n23,n24,n25,n26,n27,n28,n29,n30,n31,
2 n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n45,
3 n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60,n61,
4 n62,n63,n64,n65,n66,n67,n68,n69,n70,n71,n72,n73,n74,n75,n76,n77,
5 n78,n79,n80,n81,n82,n83,n84,locend,iname,lend
common/bk08/n4a,n4b,n4c,n4d,n4e,n4f,n4g,n4h,n7a,n7b,n7c,n7d,n7e,
1 nusir,mpusr,mpubr
COMMON/BK13/LC0,LC1H,LC1B,LC1S,LC1T,LC2,LC3,LC4,LC5,LC6,LC7,LC9,
1 LC10,LC11,LC12,LC13,LC14,LC15,LC16,LC17,LC18,LB0,LB1,LB2,
2 LC7A,LC7B
COMMON/BK20/NUMSV,JU,JV,NRTM,NRTS,NMN,NSN,NTY,NST,MST,NOCO
COMMON/BK28/SUMMSS,XKE,XPE,TT
COMMON/AUX14/SIG(49,nlq)
COMMON/SHLOPT/ISTRN,ISTUPD,IBELYT,MITER
common/sorter/nnc,lczc,
& ns11,ns12,ns13,ns14,ns15,ns16,ns17,
& nh11,nh12,nh13,nh14,nh15,nh16,nh17,
& nt11,nt12,nt13,nt14,nt15,nt16,nt17,

```



```

C      1      ,500,19)K
C  BESCHLEUNIGUNGEN
C      1 CALL KNO_ASC (ACC,1,NUMNP)
C      GOTO 490
C      2 CALL KNO_ASC (ACC,2,NUMNP)
C      GOTO 490
C      3 CALL KNO_ASC (ACC,3,NUMNP)
C      GOTO 490
C
C  REIBLEISTUNG      (noch nicht im richtigen TECPLOT-Format implementiert)
C
C  19 CONTINUE
C      KTYP=0
C      MW=1
C      LF=1
C      WRITE(93,'(3I5)') KTYP,MW,LF
C      WRITE(93,*) ' '
C      WRITE(93,*) ' '
C      DO 5291 IS=1,numnp
5291      WRITE (93,'(1X,I7,1X,E12.5)') IS, sngl(PFRIC(IS))
C      GOTO 490
C
C  E L E M E N T W E R T E
C
C  40 K=K-40
C      DUMMY=0.
C
C  HEXAHEDRONS
C
C      IF ( NUMELH .EQ. 0 ) GOTO 140
C      NELG=NUMELH/nlq
C      IF (nlq*NELG.LT.NUMELH) NELG=NELG+1
C
C      NEL=0
C      DO 100 NN=1,NELG
C      NMEL=nlq
C      IF (NN.EQ.NELG) NMEL=NUMELH-nlq*(NELG-1)
C      CALL SCALARH (A(LC1H),A(LC15),A(N1),A(NH13),A(NH14),A(NH04),
C      1          VLSTRAI,NEL,NMEL)
C
C      GOTO (41,42,100,100,100,100,100,100,100,100,100,100,100,100,100,
F      80,80,80,58,80,80,80,80,80,80,65,80,80,80,80,
F      80,80,80,80,80,80,80,80,80,80) K
C
C  HUBER-MISES-HENCKY - VERGLEICHSSPANNUNG
C
C      41 DO 410 J1=1,NMEL
C      410      WRITE (91) SIG(8,J1)
C      GOTO 100
C
C  PLAST. VERGLEICHSDDEHNUNG
C
C      42 DO 420 J1=1,NMEL
C      420      WRITE (91) SIG(7,J1)
C      GOTO 100
C  P-HYD
C      58 DO 580 J1=1,NMEL
C      580      WRITE (91) SIG(20,J1)
C      GOTO 100
C  EQUIVALENT STRAIN RATE

```

```

        65 DO 650 J1=1,NMEL
        650      WRITE (91) VLSTRAI(7,J1)
              GOTO 100
C
C   BEI GROESSEN, DIE FUER VOLUMENELEMENTE NICHT ZUR VERFUEGUNG STEHEN
C   MUESSEN BEI GEOMETRIEN MIT GEMISCHTEN ELEMNTYPEN DIESE FELDER MIT
C   NULL BELEGT WERDEN
C
        80 DO 81 J1=1,NMEL
        81      WRITE(91) DUMMY
        100 CONTINUE
C
C   BEAM ELEMENTS
C
        140 IF ( NUMELB .EQ. 0 ) GOTO 240
              NELG=NUMELB/nlq
              IF (nlq*NELG.LT.NUMELB) NELG=NELG+1
C
              NEL=0
              DO 300 NN=1,NELG
                NMEL=nlq
                IF (NN.EQ.NELG) NMEL=NUMELB-nlq*(NELG-1)
                LNS=7
                CALL SCALARB (A(NB04),A(NB13),SIG,NMEL,NEL,LNS)
C
C   !! K E I N E !!
C   HUBER-MISES-HENCKY - VERGLEICHSSPANNUNG UND
C   SPANNUNGEN IM GLOBALEN KOORD.-SYST.
C   SIG-XX, SIG-YY, SIG-ZZ, SIG-XY, SIG-YZ, SIG-ZX
C
C   PLAST. VERGLEICHSDDEHNUNG UND
C   DEHNUNGEN IM GLOBALEN KOORD.-SYST.
C   !! V O R H A N D E N !!
C
C
              GOTO (280,280,280,280,280,280,280,280,280,280,280,280,280,
F          280,280,280,280,280,280,280,280,280,280,280,280,280,280,
F          270,271,272,280,274,275,280,277,280,280) K
C
C   RESULTIERENDE SCHNITTLASTEN
C
C   MOMENT-S
        270 DO 27 J1=1,NMEL
        27      WRITE (91) SIG(4,J1)
              GOTO 300
C   MOMENT-T
        271 DO 28 J1=1,NMEL
        28      WRITE (91) SIG(5,J1)
              GOTO 300
C   TORSION
        272 DO 29 J1=1,NMEL
        29      WRITE (91) SIG(6,J1)
              GOTO 300
C   SHEAR-T
        274 DO 30 J1=1,NMEL
        30      WRITE (91) SIG(3,J1)
              GOTO 300
C   AXIAL
        275 DO 31 J1=1,NMEL
        31      WRITE (91) SIG(1,J1)

```

```

      GOTO 300
C    SHEAR-S
      277 DO 32 J1=1,NMEL
      32   WRITE (91) SIG(2,J1)
      GOTO 300
      280 DO 33 J1=1,NMEL
      33   WRITE (91) DUMMY
      300 CONTINUE
C
C    SHELL ELEMENTS
C
      240 IF ( NUMELS .EQ. 0 ) GOTO 340
      NELG=NUMELS/nlq
      IF (nlq*NELG.LT.NUMELS) NELG=NELG+1
C
      NEL=0
C *** New counter to support SCA_GET - McDermott
      nelget=0
      DO 200 NN=1,NELG
      NMEL=nlq
      IF (NN.EQ.NELG) NMEL=NUMELS-nlq*(NELG-1)
      LNS=49
      CALL SCALARS (A(N1),A(N4F),A(LC11),A(N4A),A(LC1S),A(NS05),A(NS06),
F          A(NS03),A(NS01),SIG,LNS,A(NS07),A(N4H),MPUSR,
F          A(N1+NMAT),A(NSTSL),A(NS13),A(NS14),NEL,NMEL,
F          A(NS02),EMAIN)
C
C *** Added 57 and 58 - McD '99
      GOTO (141,142,200,200,200,200,200,200,200,200,200,200,200,200,
C      F      155,156,180,180,180,180,180,180,180,180,165,180,180,180,180,
C      F      155,156,157,158,180,180,180,180,180,180,165,180,180,180,180,
C      F      170,171,172,173,174,175,176,177,178,179) K
C
C    HUBER-MISES-HENCKY - VERGLEICHSSPANNUNG UND
C    SPANNUNGEN IM GLOBALEN KOORD.-SYST.
C    SIG-XX, SIG-YY, SIG-ZZ, SIG-XY, SIG-YZ, SIG-ZX
C
      141 DO 36 J1=1,NMEL
      36   WRITE (91) SIG(8+(IINT-1)*8,J1)
      GOTO 200
C
C    PLAST. VERGLEICHSDENHNUNG UND
C    VERZERRUNGEN AN DER UNTER- UND OBERSEITE DER PLATTE
C    (IINT=1: UNTERSEITE, IINT=2: OBERSEITE) IM GLOBALEN KOORD.-SYST.
C    EPS-XX, EPS-YY, EPS-ZZ, EPS-XY, EPS-YZ, EPS-XZ,
C
C
      142 DO 37 J1=1,NMEL
      37   WRITE (91) SIG(7+(IINT-1)*8,J1)
      GOTO 200
C
C
C    principal strains EPS-I, EPS-II, strain rate
      155 DO 15 J1=1,NMEL
      WRITE (91) EMAIN(1+3*(IINT-1),J1)
      15   CONTINUE
      GOTO 200
      156 DO 16 J1=1,NMEL
      WRITE (91) EMAIN(2+3*(IINT-1),J1)
      16   CONTINUE

```

```

      GOTO 200
C
C ***** Support for Void Material (#44) - McD '99
C Void Content
157 continue
    icode=15
    call sca_get (a(n1),a(n4f),a(lc11),a(n4a),a(lc1s),a(ns06),
1      sig,lms,a(ns13),a(ns14),nelget,nmel,icode,val,iint)
    do i=1,nmel
      il = i + (nn-1)*nlq
      write(91) val(i)
    end do
C
      goto 200
C
C Yield Stress
158 continue
    icode=16
    call sca_get (a(n1),a(n4f),a(lc11),a(n4a),a(lc1s),a(ns06),
1      sig,lms,a(ns13),a(ns14),nelget,nmel,icode,val,iint)
    do i=1,nmel
      il = i + (nn-1)*nlq
      write(91) val(i)
    end do
C
      goto 200
C *****
C
165 DO 17 J1=1,NMEL
      WRITE (91) EMAIN(3+3*(IINT-1),J1)
17 CONTINUE
GOTO 200
C
C RESULTIERENDE SCHNITTLASTEN
C
C M-XX
170 DO 18 J1=1,NMEL
18 WRITE (91) SIG(25,J1)
GOTO 200
C M-YY
171 DO 14 J1=1,NMEL
14 WRITE (91) SIG(26,J1)
GOTO 200
C M-XY
172 DO 20 J1=1,NMEL
20 WRITE (91) SIG(27,J1)
GOTO 200
C Q-XX
173 DO 21 J1=1,NMEL
21 WRITE (91) SIG(28,J1)
GOTO 200
C Q-YY
174 DO 22 J1=1,NMEL
22 WRITE (91) SIG(29,J1)
GOTO 200
C N-XX
175 DO 23 J1=1,NMEL
23 WRITE (91) SIG(30,J1)
GOTO 200

```

```

C N-YY
  176 DO 24 J1=1,NMEL
  24   WRITE (91) SIG(31,J1)
      GOTO 200
C N-XY
  177 DO 25 J1=1,NMEL
  25   WRITE (91) SIG(32,J1)
      GOTO 200
C PLATTENDICKE
  178 DO 26 J1=1,NMEL
  26   WRITE (91) SIG(33,J1)
      GOTO 200
C ENERGIEDICHTE
  179 DO 34 J1=1,NMEL
  34   WRITE (91) SIG(48,J1)
      GOTO 200
C DUMMY
  180 DO 35 J1=1,NMEL
  35   WRITE (91) DUMMY
  200 CONTINUE
C
C THICK SHELLS
C
  340 IF (NUMELT.EQ.0) GOTO 440
      NELG=NUMELT/nlq
      IF (nlq*NELG.LT.NUMELT) NELG=NELG+1
C
  NEL=0
  DO 400 NN=1,NELG
    NMEL=nlq
    IF (NN.EQ.NELG) NMEL=NUMELT-nlq*(NELG-1)
    LNS=41
    CALL SCALART (A(N1),A(N4F),A(N4A),A(NT04),A(NT13),A(NT14),
  1      A(LC1T),SIG,NMEL,NEL,LNS)
C
      GOTO (441,442,400,400,400,400,400,400,479,479,479,479,479,479,
F      479,479,479,479,479,479,479,479,479,479,479,479,479,479,
F      479,479,479,479,479,479,479,479,479,479) K
C
C HUBER-MISES-HENCKY - VERGLEICHSSPANNUNG
C
  441 DO 45 J1=1,NMEL
  45   WRITE (91) SIG(35+IINT,J1)
      GOTO 400
C
  442 DO 46 J1=1,NMEL
  46   WRITE (91) SIG(7+(IINT-1)*7,J1)
      GOTO 400
C
  479 DO 47 J1=1,NMEL
  47   WRITE (91) DUMMY
  400 CONTINUE
C
  440 CONTINUE
C
C UMSCHREIBEN AUF TECPLOT-ELEMENT-SKALARFILES
C
  REWIND (91)
C
C Oeffnen und Einlesen der TECPLOT-Geometrie-Datei

```

C

```

IF (NDISP .LT. 1 .AND. NVELO .LT. 1) THEN
  GEOTIT=MOFI(1:9)///'.GEO'
ELSE IF (NDISP .GE. 1 .AND. NVELO .GE. 1) THEN
  GEOTIT=MOFI(1:9)///'.KIN'
ELSE IF (NDISP .GE. 1 .AND. NVELO .LT. 1) THEN
  GEOTIT=MOFI(1:9)///'.DIS'
ELSE IF (NDISP .LT. 1 .AND. NVELO .GE. 1) THEN
  GEOTIT=MOFI(1:9)///'.VEL'
ENDIF
OPEN(UNIT=94,FILE=GEOTIT,FORM='FORMATTED',STATUS='OLD')
READ(94,'(A80)') VARCHAR
WRITE(93,'(A80)') VARCHAR
IF( K .EQ. 1 ) THEN
  VARCHAR='VARIABLES= "X","Y","Z","SIGV"'
  WRITE(93,'(1X,A29)') VARCHAR(1:29)
ELSE IF ( K .EQ. 2 ) THEN
  VARCHAR='VARIABLES= "X","Y","Z","EPSV"'
  WRITE(93,'(1X,A29)') VARCHAR(1:29)
ELSE
  VARCHAR(1:29) ='VARIABLES= "X","Y","Z", "'//COD//"'
  WRITE(93,'(1X,A29)') VARCHAR(1:29)
ENDIF
READ(94,'(A80)') VARCHAR
READ(94,'(A80)') VARCHAR
WRITE(93,'(A80)') VARCHAR
VARCHAR = ' '
READ(94,'(A80)') VARCHAR
WRITE(93,'(A80)') VARCHAR
ELT = VARCHAR(1:27)

```

C

C Elementwerte auf Knotenwerte interpolieren

C

```

DO I = 1,NUMNP
  SS(I) = 0.
  IVAL(I) = 0
ENDDO
IEL = 0

```

C

```

IF (NUMELH.GE.1) THEN
DO 222 I = 1,NUMELH
  IEL = IEL + 1
  l=nhxpnt(i)
  READ(91) FELD
  DO 201 J2 = 1,8
    NPP = ixh(j2+1,1)
    IVAL (NPP) = IVAL(NPP) + 1
    SS(NPP) = SS(NPP) + FELD
201
222    CONTINUE
  ENDF
  IF (NUMELB.GE.1) THEN
DO 202 I = 1,NUMELB
  IEL = IEL + 1
202  READ(91) FELD
  ENDF
  IF (NUMELS.GE.1) THEN
DO 203 I = 1,NUMELS
  IEL = IEL + 1
  l=nshpnt(i)
  READ(91) FELD

```



```

DO 204 J2 = 1,3
  NPP = ixs(j2+1,1)
  IVAL (NPP) = IVAL(NPP) + 1
204  SS(NPP) = SS(NPP) + FELD
  if (ixs(4,1).ne.ixs(5,1))then
    NPP = ixs(5,1)
    IVAL (NPP) = IVAL(NPP) + 1
    SS(NPP) = SS(NPP) + FELD
  endif
203  CONTINUE
ENDIF
IF (NUMELT.GE.1) THEN
  DO 205 I = 1,NUMELT
    IEL = IEL + 1
    l=ntxpnt(i)
    READ(91,'(E12.5)') FELD
    DO 206 J2 = 1,8
      NPP = ixt(j2+1,1)
      IVAL (NPP) = IVAL(NPP) + 1
206  SS(NPP) = SS(NPP) + FELD
205  CONTINUE
    ENDIF
    DO 901 KK = 1,NUMNP
      IF (IVAL(KK).NE.0) SS(KK) = SS(KK) / FLOAT(IVAL(KK))
      READ(94,*) XX, YY, ZZ
      WRITE(93,'(4(1X,E12.5))') sngl(XX), sngl(YY), sngl(ZZ),
1      sngl(SS(KK))
901  CONTINUE
C
555  CONTINUE
    READ(94,'(A80)',END=550) VARCHAR
    WRITE(93,'(A80)') VARCHAR
    GOTO 555
C
490  CONTINUE
    CLOSE (UNIT=93)
550  CLOSE(94)
C
500  CONTINUE
C
    RETURN
C
    END

```

```

      subroutine stiffs(x,irect,stf,bh,ipss,cm,matype,eosp,ieost,numelh,
1numels,numelt,nrt,nmmat,ro,zf,thicks,ethik,nty,chrln,ipst,
1jxl,jx7,nsv,nsn,stfvg,thk,sftr,ishlfm,itytp)
      implicit double precision (a-h,o-z)
      dp
c
c
c called by initlz to compute bulk modulus of each material for sliding
c interface stiffness determination
c
c      call stiffs (x,irects(k1),stfs(k9),ipsh,b(lcls),cm,matype,eosp,
c      1 ieost,numelh,numels,numelt,nrts,nmmat,ro,zfcs,b(ns05),ethik(k4),
c      2 nty,chrln(n),b(lclt),b(ibins1),b(ibins2),nsv(k4),nsn,stfss,
c      3 thkslv(k9),b(nlcslv),b(nl+nmmat),itytp)
c
c
c
c
c *** Added Mat #44 (Elasto-Plastic with Void Effects) - McDermott 1999
c
c
c      if (mt.eq.41)bkm(mx)=cm(mx48m1+21)
c Elasto-Plastic with Void Effects - McD '99
c      if (mt.eq.44)bkm(mx)=cm(mx48m1+1)/(3.*(1.-2.*cm(mx48m1+2)))
c
c
c
c
end

```

```

      subroutine stifsn(x,irect,stf,ipsh,ipss,cm,matype,eosp,ieost,
1numelh,numels,numelt,nrt,nmmat,ro,zf,thicks,ethik,nty,chrln,ipst,
1jxl,jx7,nsv,nsn,stfvg,thk,sftr,ihlsnd,islsnd,isg2el,stfsnd,
1thksnd,ishltp,ishlfm)
      implicit double precision (a-h,o-z)
      dp
c
c called by initlz to compute bulk modulus for each material
c to determine sliding interface stiffness with materials in
c master sand volume
c
c   call stifsn (x,irectm(k6),stfm(k10),ipsh,b(lcls),cm,mttype,eosp,
c   1 ieost,numelh,numels,numelt,nrtm,nmmat,ro,zfcm,b(ns05),fthik(k5),
c   2 nty,chrln(n),b(lclt),b(ibins1),b(ibins2),msr(k5),nmn,stfsm,
c   3 thkmsr(k10),b(nlcmsr),ihlsnd(kd3),islsnd(kd4),
c   4 isg2el(kd7),stfsnd(kd11),thksnd(kd12),ishltp(kd5),b(n1+nmmat))
c
c slave sand volume
c
c   call stifsn (x,irects(k1),stfs(k9),ipsh,b(lcls),cm,mttype,eosp,
c   1 ieost,numelh,numels,numelt,nrts,nmmat,ro,zfcs,b(ns05),ethik(k4),
c   2 nty,chrln(n),b(lclt),b(ibins1),b(ibins2),nsv(k4),nsn,stfss,
c   3 thkslv(k9),b(nlcslv),ihlsnd(kd3),islsnd(kd4),
c   4 isg2el(kd7),stfsnd(kd11),thksnd(kd12),ishltp(kd5),b(n1+nmmat))
c
c *** Added Elasto-Plastic with Void Effects (#44) - McDermott 1999
c
c
c
c   if (mt.eq.39) bkm(mx)=cm(mx48ml+1)/(3.*(1.-2.*cm(mx48ml+2)))
c *****
c Elasto-Plastic with Void Effects - McD '99
c   if (mt.eq.44) bkm(mx)=cm(mx48ml+1)/(3.*(1.-2.*cm(mx48ml+2)))
c *****
c
c
c
c end

```

```

SUBROUTINE TEC_TEN (A, nhxpnt,nshpnt,ntxpnt,ixh,ixs,ixt,ss,ival )
C
C WRITE SCALAR PLOT FILES (TECPLOT-FORMAT) ONLY STRESS- AND STRAIN-
C TENSOR
C REIHENFOLGE WICHTIG !!!
C 1.) VOLUMENELEMENTE
C 2.) BALKENELEMENTE (werden von TECPLOT
nicht unterstuetzt)
C 3.) SCHALENELEMENTE
C 4.) DICKE SCHALENELEMENTE
C
C Corrected bug in some of the scratchfile writes - they must be
C the same precision and the variable used for reads (ie double)
C (Did not mark changes, since they occur throughout subroutine)
C - McDermott, '99
C*****
implicit double precision (a-h,o-z)
dp
include 'nlqpar.inc'
COMMON/BK00/NUMNP, NUMPC, NUMLP, NEQ, NDOF, NLCUR, NUMCL, NUMVC,
1 NDTPTS, NELMD, NMMAT, NUMELH, NUMELB, NUMELS, NUMELT, NUMDP,
2 GRVITY, IDIRGV, NODSPC, NSPCOR
cfu
common/bk03/endtim,prtc,pltc,ndthl,nsthl,nstsl,nstbl,nsttl,mkthf
common/bk03/endtim,prtc,pltc,ngthl,ndthl,nsthl,nstsl,nstbl,
1 nsttl,ncpll,mkthf
COMMON/BK04/PRTOUT,PLTOUT,DT2OLD,SLSFAC,TSSFAC,IHYDRO,
cfu 1 NDTH,NMST,NSTH,NSTS,NSTB,NSTT,IKEDIT
1 ngth,ndth,nmst,nsth,nsts,nstb,nstt,ncpl,ikedit
COMMON/BK05/
1 NH01,NH02,NH03,NH04,NH05,NH06,NH07,NH08,NH09,NH10,
2 NB01,NB02,NB03,NB04,NB05,NB06,NB07,NB08,NB09,NB10,
3 NS01,NS02,NS03,NS04,NS05,NS06,NS07,NS08,NS09,NS10,
4 NT01,NT02,NT03,NT04,NT05,NT06,NT07,NT08,NT09,NT10
REAL*8 HEAD
VAX750
common/bk06/time(2,8),head(12),idmmy,iadd,ifil,maxsiz,ncycle
common/bk07/n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11,n12,n13,n14,n15,
1 n16,n17,n18,n19,n20,n21,n22,n23,n24,n25,n26,n27,n28,n29,n30,n31,
2 n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n45,
3 n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60,n61,
4 n62,n63,n64,n65,n66,n67,n68,n69,n70,n71,n72,n73,n74,n75,n76,n77,
5 n78,n79,n80,n81,n82,n83,n84,locend,iname,lendf
common/bk08/n4a,n4b,n4c,n4d,n4e,n4f,n4g,n4h,n7a,n7b,n7c,n7d,n7e,
1 nusir,mpusr,mpubr
COMMON/BK13/LC0,LC1H,LC1B,LC1S,LC1T,LC2,LC3,LC4,LC5,LC6,LC7,LC9,
1 LC10,LC11,LC12,LC13,LC14,LC15,LC16,LC17,LC18,LB0,LB1,LB2,
2 LC7A,LC7B
COMMON/BK20/NUMSV,JU,JV,NRTM,NRTS,NMN,NSN,NTY,NST,MST,NOCO
COMMON/BK28/SUMMSS,XKE,XPE,TT
COMMON/AUX14/SIG(49,nlq)
COMMON/SHLOPT/ISTRN,ISTUPD,IBELYT,MITER
common/sorter/nnc,lczc,
& ns11,ns12,ns13,ns14,ns15,ns16,ns17,
& nh11,nh12,nh13,nh14,nh15,nh16,nh17,
& nt11,nt12,nt13,nt14,nt15,nt16,nt17,
& nb11,nb12,nb13,nb14,nb15,nb16,nb17
COMMON/INKTH/ INCHIS, NPOST, NDISP, NVELO, NCYREM,
1 NSTRESS, NSTRAIN

```

```

CHARACTER*8 DATU,VS
CHARACTER*60 SBTEXT(20)
COMMON/VSNUM/VS,DATU
CHARACTER MOFI*13,COD*3,COD1*6, GEOTIT*13
CHARACTER VARCHAR*84, ELT*27
C
DIMENSION A(*)
DIMENSION VLSTRAI(7,nlq), EMAIN(6,nlq)
DIMENSION SS(*), FELD(14), VAL1(7), IVAL(*)
dimension ixh(9,*),ixs(5,*),ixt(9,*),
dimension nhxpnt(*),nshpnt(*),ntxpnt(*)
C
C
WRITE (COD1,'(I6)') NCYCLE
C
IF (NCYCLE.LT.1000000) COD1(1:1)='0'
IF (NCYCLE.LT.100000) COD1(2:2)='0'
IF (NCYCLE.LT.10000) COD1(3:3)='0'
IF (NCYCLE.LT.1000) COD1(4:4)='0'
IF (NCYCLE.LT.100) COD1(5:5)='0'
C
MOFI(1:3) = 'TEC'
MOFI(4:9) = COD1
DUMMY=0.
C
IF (NSTRESS .EQ. 1) THEN
C
COD = 'Sij'
MOFI(10:13)='.'//COD
C
OPEN (UNIT=91,POSITION='REWIND',
1 STATUS='SCRATCH',FORM='UNFORMATTED')
OPEN (UNIT=93,STATUS='UNKNOWN',FILE=MOFI,FORM='FORMATTED')
C
C
C
HEXAHEDRONS
C
IF ( NUMELH .EQ. 0 ) GOTO 140
NELG=NUMELH/nlq
IF (nlq*NELG.LT.NUMELH) NELG=NELG+1
C
NEL=0
DO 100 NN=1,NELG
NMEL=nlq
IF (NN.EQ.NELG) NMEL=NUMELH-nlq*(NELG-1)
CALL SCALARH (A(LC1H),A(LC15),A(N1),A(NH13),A(NH14),A(NH04),
1 VLSTRAI,NEL,NMEL)
C
C
C
HUBER-MISES-HENCKY - VERGLEICHSSPANNUNG UND
C SPANNUNGEN IM GLOBALEN KOORD.-SYST.
C SIG-XX, SIG-YY, SIG-ZZ, SIG-XY, SIG-YZ, SIG-ZX
C
DO J1=1,NMEL
WRITE (91) SIG(8,J1),SIG(1,J1),SIG(2,J1),SIG(3,J1),
1 SIG(4,J1),SIG(6,J1),SIG(5,J1),
2 SIG(8,J1),SIG(1,J1),SIG(2,J1),SIG(3,J1),
3 SIG(4,J1),SIG(6,J1),SIG(5,J1)
ENDDO
C
100 CONTINUE
C

```

```

C BEAM ELEMENTS
C
140 IF ( NUMELB .EQ. 0 ) GOTO 240
    NELG=NUMELB/nlq
    IF (nlq*NELG.LT.NUMELB) NELG=NELG+1
C
    NEL=0
    DO 300 NN=1,NELG
        NMEL=nlq
        IF (NN.EQ.NELG) NMEL=NUMELB-nlq*(NELG-1)
C
        LNS=7
C
        CALL SCALARB (A(NB04),A(NB13),SIG,NMEL,NEL,LNS)
        DO J1=1,NMEL
            WRITE (91) ( DUMMY, J14=1,14 )
        ENDDO
    300 CONTINUE
C
C SHELL ELEMENTS
C
240 IF ( NUMELS .EQ. 0 ) GOTO 340
    NELG=NUMELS/nlq
    IF (nlq*NELG.LT.NUMELS) NELG=NELG+1
C
    NEL=0
    DO 200 NN=1,NELG
        NMEL=nlq
        IF (NN.EQ.NELG) NMEL=NUMELS-nlq*(NELG-1)
        LNS=49
        CALL SCALARS (A(N1),A(N4F),A(LC11),A(N4A),A(LC1S),A(NS05),A(NS06)
F            ,A(NS03),A(NS01),SIG,LNS,A(NS07),A(N4H),MPUSR,
F            A(N1+NMAT),A(NSTSL),A(NS13),A(NS14),NEL,NMEL,
F            A(NS02),EMAIN)
C
C HUBER-MISES-HENCKY - VERGLEICHSSPANNUNG UND
C SPANNUNGEN IM GLOBALEN KOORD.-SYST.
C SIG-XX, SIG-YY, SIG-ZZ, SIG-XY, SIG-YZ, SIG-ZX
C
        DO J1=1,NMEL
            WRITE (91) SIG(16,J1),SIG(9,J1),SIG(10,J1),SIG(11,J1),
1                SIG(12,J1),SIG(14,J1),SIG(13,J1),SIG(24,J1),
2                SIG(17,J1),SIG(18,J1),SIG(19,J1),SIG(20,J1),
3                SIG(22,J1),SIG(21,J1)
        ENDDO
    200 CONTINUE
C
C THICK SHELLS
C
340 IF (NUMELT.EQ.0) GOTO 440
    NELG=NUMELT/nlq
    IF (nlq*NELG.LT.NUMELT) NELG=NELG+1
C
    NEL=0
    DO 400 NN=1,NELG
        NMEL=nlq
        IF (NN.EQ.NELG) NMEL=NUMELT-nlq*(NELG-1)
        LNS=41
        CALL SCALART (A(N1),A(N4F),A(N4A),A(NT04),A(NT13),A(NT14),
1                A(LC1T),SIG,NMEL,NEL,LNS)
C
C HUBER-MISES-HENCKY - VERGLEICHSSPANNUNG UND

```

```

C SPANNUNGEN IM GLOBALEN KOORD.-SYST.
C SIG-XX, SIG-YY, SIG-ZZ, SIG-XY, SIG-YZ, SIG-ZX
C
  DO J1=1,NMEL
    WRITE (91) SIG(37,J1),SIG(8,J1),SIG(9,J1),SIG(10,J1),
1      SIG(11,J1),SIG(13,J1),SIG(12,J1),SIG(38,J1),SIG(15,J1),
2      SIG(16,J1),SIG(17,J1),SIG(18,J1),SIG(20,J1),
3      SIG(19,J1)
  ENDDO
C
400 CONTINUE
440 CONTINUE
C
C UMSCHREIBEN AUF TECPLOT-ELEMENT-SKALARFILES
C
  REWIND (91)
C
C Oeffnen und Einlesen der TECPLOT-Geometrie-Datei
C
  IF (NDISP .LT. 1 .AND. NVELO .LT. 1) THEN
    GEOTIT=MOFI(1:9)//'.GEO'
  ELSE IF (NDISP .GE. 1 .AND. NVELO .GE. 1) THEN
    GEOTIT=MOFI(1:9)//'.KIN'
  ELSE IF (NDISP .GE. 1 .AND. NVELO .LT. 1) THEN
    GEOTIT=MOFI(1:9)//'.DIS'
  ELSE IF (NDISP .LT. 1 .AND. NVELO .GE. 1) THEN
    GEOTIT=MOFI(1:9)//'.VEL'
  ENDIF
  OPEN(UNIT=94,FILE=GEOTIT,FORM='FORMATTED',STATUS='OLD')
  READ(94,'(A80)') VARCHAR
  WRITE(93,'(A80)') VARCHAR
  VARCHAR='VARIABLES= "X","Y","Z","SIGV","SIG-XX","SIG-YY"//'
1  VARCHAR=VARCHAR||'"SIG-ZZ","SIG-XY","SIG-YZ","SIG-ZX"'
  WRITE(93,'(1X,A84)') VARCHAR(1:84)
  READ(94,'(A80)') VARCHAR
  READ(94,'(A80)') VARCHAR
  WRITE(93,'(A80)') VARCHAR
  VARCHAR = ' '
  READ(94,'(A80)') VARCHAR
  WRITE(93,'(A80)') VARCHAR
  ELT = VARCHAR(1:27)
C
C Elementwerte auf Knotenwerte interpolieren
C
C Spannungstensor
C
  DO 880 L = 1,2
    DO 800 K = (L-1)*7+1 , (L-1)*7+7
      REWIND(91)
      IEL = 0
      DO I = 1,NUMNP
        SS(I) = 0.
        IVAL(I) = 0
      ENDDO
      IF (NUMELH.GE.1) THEN
        DO 600 I = 1,NUMELH
          IEL = IEL + 1
          ll=nhxpnt(i)
          READ(91) (FELD(II),II=1,14)
          DO 700 J2 = 1,8

```

```

        NPP = ixh(j2+1,11)
        IVAL (NPP) = IVAL(NPP) + 1
700      SS(NPP) = SS(NPP) + FELD(K)
600      CONTINUE
      ENDIF
C
      IF (NUMELB.GE.1) THEN
      DO 601 I = 1,NUMELB
        IEL = IEL + 1
601      READ(91) FELD(1)
      ENDIF
C
      IF (NUMELS.GE.1) THEN
      DO 602 I = 1,NUMELS
        IEL = IEL + 1
        ll=nshpnt(i)
        READ(91) (FELD(II),II=1,14)
        DO 702 J2 = 1,3
          NPP = ixh(j2+1,11)
          IVAL (NPP) = IVAL(NPP) + 1
702      SS(NPP) = SS(NPP) + FELD(K)
          if (ixs(4,ll).ne.ixs(5,ll))then
            NPP = ixh(5,ll)
            IVAL (NPP) = IVAL(NPP) + 1
            SS(NPP) = SS(NPP) + FELD(K)
          endif
602      CONTINUE
      ENDIF
C
      IF (NUMELT.GE.1) THEN
      DO 603 I = 1,NUMELT
        IEL = IEL + 1
        ll=ntxpnt(i)
        READ(91) (FELD(II),II=1,14)
        DO 703 J2 = 1,8
          NPP = ixt(j2+1,11)
          IVAL (NPP) = IVAL(NPP) + 1
703      SS(NPP) = SS(NPP) + FELD(K)
603      CONTINUE
      ENDIF
C
      IOP = 23+K-(L-1)*7
      OPEN(UNIT=IOP,STATUS='SCRATCH',FORM='UNFORMATTED')
      DO 900 KK = 1,NUMNP
        IF (IVAL(KK).NE.0) SS(KK) = SS(KK) / FLOAT(IVAL(KK))
900      WRITE(IOP) SS(KK)
800      CONTINUE
C
      DO 910 I=1,7
        IOP = 23+I
910      REWIND(IOP)
C
      IF (L.EQ.2) THEN
        VARCHAR(1:50) =ELT/' D=(1,2,3,FECONNECT)
        WRITE(93,'(A50)') VARCHAR
        DO 850 I = 1,NUMNP
          DO 860 II = 1,7
            IOP=23+II
860          READ(IOP) VAL1(II)
850          WRITE(93,'(7(1X,E12.5))') (sngl(VAL1(IK)),IK=1,7)

```



```

        GOTO 999
    ENDIF
C
    DO 810 I = 1, NUMNP
        READ(94,*) XX, YY, ZZ
        DO 820 II = 1, 7
            IOP=23+II
            READ(IOP) VAL1(II)
820        CONTINUE
            WRITE(93, '(10(1X,E12.5))') XX, YY, ZZ, (VAL1(IK), IK=1, 7)
810        CONTINUE
C
        DO 830 I = 1, 7
            IOP = 23+I
830        CLOSE(IOP)
C
888    CONTINUE
        READ(94, '(A80)', END=880) VARCHAR
        ICOL = INDEX(VARCHAR, 'D=')
        IF (ICOL.GT.0) VARCHAR(ICOL:ICOL+24) = 'D=(1,2,3,4,5,6,7,8,9,10)'
        WRITE(93, '(A80)') VARCHAR
        GOTO 888
880    CONTINUE
C
        CLOSE (93)
999    CLOSE (94)
C
    ENDIF
C
    IF (NSTRAIN .EQ. 1 ) THEN
C
        COD = 'Eij'
        MOFI(10:13)='.'//COD
C
        OPEN (UNIT=91, POSITION='REWIND',
1          STATUS='SCRATCH', FORM='UNFORMATTED')
        OPEN (UNIT=93, STATUS='UNKNOWN', FILE=MOFI, FORM='FORMATTED')
C
C    HEXAHEDRONS
C
        IF ( NUMELH .EQ. 0 ) GOTO 1140
        NELG=NUMELH/nlq
        IF (nlq*NELG.LT.NUMELH) NELG=NELG+1
C
        NEL=0
        DO 1100 NN=1, NELG
            NMEL=nlq
            IF (NN.EQ.NELG) NMEL=NUMELH-nlq*(NELG-1)
            CALL SCALARH (A(LC1H), A(LC15), A(N1), A(NH13), A(NH14), A(NH04),
1          VLSTRAI, NEL, NMEL)
C
C    PLAST. VERGLEICHSDDEHNUNG UND
C    DEHNUNGEN IM GLOBALEN KOORD.-SYST.
C    EPS-XX, EPS-YY, EPS-ZZ, EPS-XY, EPS-YZ, EPS-XZ,
C
        DO J1=1, NMEL
            WRITE (91) SIG(7, J1), VLSTRAI(1, J1), VLSTRAI(2, J1), VLSTRAI(3, J1),
1          VLSTRAI(4, J1), VLSTRAI(6, J1), VLSTRAI(5, J1),
2          SIG(7, J1), VLSTRAI(1, J1), VLSTRAI(2, J1), VLSTRAI(3, J1),
3          VLSTRAI(4, J1), VLSTRAI(6, J1), VLSTRAI(5, J1)

```

```

      ENDDO
1100 CONTINUE
C
C   BEAM ELEMENTS
C
1140 IF ( NUMELB .EQ. 0 ) GOTO 1240
      NELG=NUMELB/nlq
      IF (nlq*NELG.LT.NUMELB) NELG=NELG+1
C
      NEL=0
      DO 1300 NN=1,NELG
        NMEL=nlq
        IF (NN.EQ.NELG) NMEL=NUMELB-nlq*(NELG-1)
C
C       LNS=7
C       CALL SCALARB (A(NB04),A(NB13),SIG,NMEL,NEL,LNS)
        DO J1=1,NMEL
          WRITE (91) ( DUMMY, J14=1,14 )
        ENDDO
1300 CONTINUE
C
C   SHELL ELEMENTS
C
1240 IF ( NUMELS .EQ. 0 ) GOTO 1340
      NELG=NUMELS/nlq
      IF (nlq*NELG.LT.NUMELS) NELG=NELG+1
C
      NEL=0
      DO 1350 NN=1,NELG
        NMEL=nlq
        IF (NN.EQ.NELG) NMEL=NUMELS-nlq*(NELG-1)
        LNS=49
        CALL SCALARS (A(N1),A(N4F),A(LC11),A(N4A),A(LC1S),A(NS05),A(NS06)
F          ,A(NS03),A(NS01),SIG,LNS,A(NS07),A(N4H),MPUSR,
F          A(N1+NMAT),A(NSTSL),A(NS13),A(NS14),NEL,NMEL,
F          A(NS02),EMAIN)
C
C   PLAST. VERGLEICHSDEHNUNG UND
C   VERZERRUNGEN AN DER UNTER- UND OBERSEITE DER PLATTE
C   (IINT=1: UNTERSEITE, IINT=2: OBERSEITE) IM GLOBALEN KOORD.-SYST.
C   EPS-XX, EPS-YY, EPS-ZZ, EPS-XY, EPS-YZ, EPS-XZ,
C
      DO J1=1,NMEL
        WRITE (91) SIG(15,J1),SIG(36,J1),SIG(37,J1),SIG(38,J1),
1          SIG(39,J1),SIG(41,J1),SIG(40,J1),SIG(23,J1),
2          SIG(42,J1),SIG(43,J1),SIG(44,J1),SIG(45,J1),
3          SIG(47,J1),SIG(46,J1)
      ENDDO
1350 CONTINUE
C
C   THICK SHELLS
C
1340 IF (NUMELT.EQ.0) GOTO 1540
      NELG=NUMELT/nlq
      IF (nlq*NELG.LT.NUMELT) NELG=NELG+1
C
      NEL=0
      DO 1500 NN=1,NELG
        NMEL=nlq
        IF (NN.EQ.NELG) NMEL=NUMELT-nlq*(NELG-1)
        LNS=41

```

```

      CALL SCALART (A(N1),A(N4F),A(N4A),A(NT04),A(NT13),A(NT14),
1      A(LC1T),SIG,NMEL,NEL,LNS)
C
      DO J1=1,NMEL
        WRITE (91) SIG(14,J1), (DUMMY, I=1,6),
1      SIG(21,J1), (DUMMY, I=1,6)
      ENDDO
1500 CONTINUE
1540 CONTINUE
C
C  UMSCHREIBEN AUF TECPLOT-ELEMENT-SKALARFILES
C
      REWIND (91)
C
C  Oeffnen und Einlesen der TECPLOT-Geometrie-Datei
C
      IF (NDISP .LT. 1 .AND. NVELO .LT. 1) THEN
        GEOTIT=MOFI(1:9)//'.GEO'
      ELSE IF (NDISP .GE. 1 .AND. NVELO .GE. 1) THEN
        GEOTIT=MOFI(1:9)//'.KIN'
      ELSE IF (NDISP .GE. 1 .AND. NVELO .LT. 1) THEN
        GEOTIT=MOFI(1:9)//'.DIS'
      ELSE IF (NDISP .LT. 1 .AND. NVELO .GE. 1) THEN
        GEOTIT=MOFI(1:9)//'.VEL'
      ENDIF
      OPEN(UNIT=94,FILE=GEOTIT,FORM='FORMATTED',STATUS='OLD')
      READ(94,'(A80)') VARCHAR
      WRITE(93,'(A80)') VARCHAR
      VARCHAR='VARIABLES= "X","Y","Z","EPSV","E-XX","E-YY","E-ZZ"//'
1      VARCHAR=VARCHAR//", "E-XY","E-YZ","E-ZX"'"
      WRITE(93,'(1X,A72)') VARCHAR(1:72)
      READ(94,'(A80)') VARCHAR
      READ(94,'(A80)') VARCHAR
      WRITE(93,'(A80)') VARCHAR
      VARCHAR = ' '
      READ(94,'(A80)') VARCHAR
      WRITE(93,'(A80)') VARCHAR
      ELT = VARCHAR(1:27)
C
C  Elementwerte auf Knotenwerte interpolieren
C
C  Dehnungstensor
C
      DO 1880 L = 1,2
        DO 1800 K = (L-1)*7+1 , (L-1)*7+7
          REWIND(91)
          IEL = 0
          DO I = 1,NUMNP
            SS(I) = 0.
            IVAL(I) = 0
          ENDDO
          IF (NUMELH.GE.1) THEN
            DO 1600 I = 1,NUMELH
              IEL = IEL + 1
              ll=nhxpnt(i)
              READ(91) (FELD(II),II=1,14)
              DO 1700 J2 = 1,8
                NPP = ixh(j2+1,ll)
                IVAL (NPP) = IVAL(NPP) + 1
                SS(NPP) = SS(NPP) + FELD(K)
1700

```

```

1600      CONTINUE
      ENDIF
C
      IF (NUMELB.GE.1) THEN
      DO 1601 I = 1,NUMELB
        IEL = IEL + 1
1601      READ(91) FELD(1)
      ENDIF
C
      IF (NUMELS.GE.1) THEN
      DO 1602 I = 1,NUMELS
        IEL = IEL + 1
        ll=nshpnt(i)
        READ(91) (FELD(II),II=1,14)
        DO 1702 J2 = 1,3
          NPP = ixS(j2+1,ll)
          IVAL (NPP) = IVAL(NPP) + 1
1702      SS(NPP) = SS(NPP) + FELD(K)
          if (ixS(4,ll).ne.ixS(5,ll))then
            NPP = ixS(5,ll)
            IVAL (NPP) = IVAL(NPP) + 1
            SS(NPP) = SS(NPP) + FELD(K)
          endif
1602      CONTINUE
      ENDIF
C
      IF (NUMELT.GE.1) THEN
      DO 1603 I = 1,NUMELT
        IEL = IEL + 1
        ll=ntxpnt(i)
        READ(91) (FELD(II),II=1,14)
        DO 1703 J2 = 1,8
          NPP = ixt(j2+1,ll)
          IVAL (NPP) = IVAL(NPP) + 1
1703      SS(NPP) = SS(NPP) + FELD(K)
1603      CONTINUE
      ENDIF
C
      IOP = 23+K-(L-1)*7
      OPEN(UNIT=IOP,POSITION='REWIND',
1      STATUS='SCRATCH',FORM='UNFORMATTED')
      DO 1900 KK = 1,NUMNP
        IF (IVAL(KK).NE.0) SS(KK) = SS(KK) / FLOAT(IVAL(KK))
1900      WRITE(IOP),SS(KK)
1800      CONTINUE
C
      DO 1910 I=1,7
        IOP = 23+I
1910      REWIND(IOP)
C
      IF (L.EQ.2) THEN
        VARCHAR(1:50) =ELT//' D=(1,2,3,FECONNECT)
        WRITE(93,'(A50)') VARCHAR
        DO 1850 I = 1,NUMNP
          DO 1860 II = 1,7
            IOP=23+II
1860      READ(IOP) VAL1(II)
1850      WRITE(93,'(7(1X,E12.5))') (VAL1(IK),IK=1,7)
          GOTO 2550
        ENDIF

```

```

C
    DO 1810 I = 1, NUMNP
        READ(94,*) XX, YY, ZZ
        DO 1820 II = 1, 7
            IOP=23+II
            READ(IOP) VAL1(II)
1820        CONTINUE
            WRITE(93, '(10(1X,E12.5))') XX, YY, ZZ, (VAL1(IK), IK=1, 7)
1810        CONTINUE
C
        DO 1830 I = 1, 7
            IOP = 23+I
1830        CLOSE(IOP)
C
1888    CONTINUE
        READ(94, '(A80)', END=1880) VARCHAR
        ICOL = INDEX(VARCHAR, 'D=')
        IF (ICOL.GT.0) VARCHAR(ICOL:ICOL+24) = 'D=(1,2,3,4,5,6,7,8,9,10)'
        WRITE(93, '(A80)') VARCHAR
        GOTO 1888
1880    CONTINUE
C
        CLOSE (93)
2550    CLOSE (94)
        ENDIF
C
        RETURN
        END

```

```

      SUBROUTINE TECPLOT ( V, X, X0,nhxpnt,nshpnt,ntxpnt,ixh,ixs,ixt)
C
C   This SBR writes for every flagged cycle the actual node
configuration
C   of the whole structure in TECPLOT-format (ASCII)
C
C   Added global variable to keep track of number of files (TECPLOT
zones),
C   and attach cycle and time text to the zone, removed a couple
associated
C   character variables
C   Not complete, since it doesn't work for files where there are
C   two zones in one file (Like stress and strain) - McDermott '99
C*****
      implicit double precision (a-h,o-z)
      dp
      REAL*8 HEAD
VAX750
      COMMON/BK00/NUMNP,NUMPC,NUMLP,NEQ,NDOF,NLCUR,NUMCL,NUMVC,
1      NDTPTS,NELMD,NMMAT,NUMELH,NUMELB,NUMELS,NUMELT,NUMDP,
2      GRVITY,IDIRGV,NODSPC,NSPCOR
      common/bk06/time(2,8),head(12),idmmy,iadd,ifil,maxsiz,ncycle
      COMMON/BK28/SUMMSS,XKE,XPE,TT
      COMMON/INKTH/ INCHIS, NPOST, NDISP, NVELO, NCYREM,
1      NSTRESS, NSTRAIN
      COMMON/ELTYP/ NEV
C *** New Variable for current TECPLOT zone number - McD
      common/TEC/ ize
C   integer save ize
      CHARACTER*8 DATU
      COMMON/VNUM/VN,DATU
      CHARACTER*24 DISCHAR , XYCHAR
      CHARACTER*18 VELCHAR
      CHARACTER*13 MOFI
      CHARACTER*6 COD
      CHARACTER VARCHAR*66, VN*8, DAT*9
      character*66 varchal
C   character*66 TEXT
      DIMENSION X(3,*), X0(3,*), V(3,*)
      dimension ixh(9,*),ixs(5,*),ixt(9,*)
      dimension nhxpnt(*),nshpnt(*),ntxpnt(*)
C
      DATA XYCHAR /'VARIABLES= "X", "Y", "Z"/
      DATA VELCHAR /', "VX", "VY", "VZ"/
      DATA DISCHAR /', "DISX", "DISY", "DISZ"/
C
C *** Initialize ize of first cycle
      if (ncycle.le.1) ize = 0
      ize = ize + 1

      WRITE (COD,'(I6)') NCYCLE
      IF (NCYCLE.LT.100000) COD(1:1)='0'
      IF (NCYCLE.LT.10000) COD(2:2)='0'
      IF (NCYCLE.LT.1000) COD(3:3)='0'
      IF (NCYCLE.LT.100) COD(4:4)='0'
      IF (NCYCLE.LT.10) COD(5:5)='0'
      MOFI(1:3) = 'TEC'
      MOFI(4:9) = COD
      IF (NDISP .LT. 1 .AND. NVELO .LT. 1) THEN
        MOFI(10:13)='GEO'

```

```

        IDUP = 1
    ELSE IF (NDISP .GE. 1 .AND. NVELO .GE. 1) THEN
        MOFI(10:13)='.KIN'
        IDUP = 3
    ELSE IF (NDISP .GE. 1 .AND. NVELO .LT. 1) THEN
        MOFI(10:13)='.DIS'
        IDUP = 2
    ELSE IF (NDISP .LT. 1 .AND. NVELO .GE. 1) THEN
        MOFI(10:13)='.VEL'
        IDUP = 2
    ENDIF

C
    CALL DATE(DAT)
C
C   Oeffnen und schreiben der TECPLOT Datei
C
C
    NKN=NUMNP
    IZO = 0
    KHILF = 0
    VARCHAR = XYCHAR
    IC = 24

C
    OPEN (UNIT=31,STATUS='UNKNOWN',FILE=MOFI(1:13),FORM='FORMATTED')
C
    IF ( NDISP .GE. 1 ) THEN
        VARCHAR = XYCHAR//DISCHAR
        IC = 48
    ENDIF
    IF ( NVELO .GE. 1 ) then
        VARCHA1 = VARCHAR(1:IC)//VELCHAR
        varchar =varcha1
    endif

C
c *** Changed to attach text string to current zone - McD
c   TEXT='TEXT X=0.14,Y=0.9,T="CYCLE '//ACYCLE//', TIME '//ATIM//'
S"'
    WRITE(31,*) 'TITLE = " DYNA_N - VERSION ',VS,' ',DAT,' "'
    WRITE(31,*) VARCHAR
c   WRITE(31,*) TEXT
    write(31,203) ize,ncycle,tt
    IF ( NUMELB + NUMELS .EQ. NEV) THEN
        IF (NUMELS .GE. 1) WRITE(31,201) NKN, NUMELS
    ELSE
        WRITE(31,202) NKN, NUMELH
    ENDIF
201  FORMAT(1X,'ZONE F=FEPOINT, N=',I7,', E=',I7,', ET=QUADRILATERAL')
202  FORMAT(1X,'ZONE F=FEPOINT, N=',I7,', E=',I7,', ET=BRICK')
203  format(1x,'TEXT X=0.14,Y=0.9,ZN=',i6,', T="CYCLE ',i6,', TIME ',
1      E14.5,' SEC"')

C
C   Knotenkoordinaten, + evtl. Displacements und Velocities
C
    DO 4711 K = 1,NKN
        IF (NDISP .GE. 1 .AND. NVELO .GE. 1)
1      WRITE(31,'(9(1X,E12.5))')
2          sngl(X(1,K)), sngl(X(2,K)), sngl(X(3,K)),
3          sngl(X(1,K)-X0(1,K)), sngl(X(2,K)-X0(2,K)),
4          sngl(X(3,K)-X0(3,K)),
5          sngl(V(1,K)), sngl(V(2,K)), sngl(V(3,K))

```

```

      IF (NDISP .GE. 1 .AND. NVELO .LT. 1)
1    WRITE(31, '(6(1X,E12.5))')
2      sngl(X(1,K)), sngl(X(2,K)), sngl(X(3,K)),
3      sngl(X(1,K)-X0(1,K)), sngl(X(2,K)-X0(2,K)),
4      sngl(X(3,K)-X0(3,K))
      IF (NDISP .LT. 1 .AND. NVELO .GE. 1)
1    WRITE(31, '(6(1X,E12.5))')
2      sngl(X(1,K)), sngl(X(2,K)), sngl(X(3,K)),
3      sngl(V(1,K)), sngl(V(2,K)), sngl(V(3,K))
      IF (NDISP .LT. 1 .AND. NVELO .LT. 1)
1    WRITE(31, '(3(1X,E12.5))')
2      sngl(X(1,K)), sngl(X(2,K)), sngl(X(3,K))
4711 CONTINUE
C
C ELEMENTCONNECTIVITY aufgeteilt in ZONES (je EL-TYP eine ZONE)
C
      IF ( NUMELH .GE. 1 ) THEN
        IZO = IZO + 1
        DO 4712 N=1,NUMELH
          l=nhxpnt(n)
          WRITE(31, '(8I8)') (ixh(i,l), i=2,9)
4712 CONTINUE
        ENDIF
        IEL = NUMELH + NUMELB
C
      IF (NUMELS .GE. 1) THEN
C
        IF (IZO.GE.1.AND.IDUP.EQ.1) WRITE(31,401) NKN, NUMELS
        IF (IZO.GE.1.AND.IDUP.EQ.2) WRITE(31,402) NKN, NUMELS
        IF (IZO.GE.1.AND.IDUP.EQ.3) WRITE(31,403) NKN, NUMELS
401      FORMAT(1X, 'ZONE F=FEPOINT, N=', I7, ', E=', I7,
1          ', ET=QUADRILATERAL, D=(1,2,3)')
402      FORMAT(1X, 'ZONE F=FEPOINT, N=', I7, ', E=', I7,
F          ', ET=QUADRILATERAL, D=(1,2,3,4,5,6)')
403      FORMAT(1X, 'ZONE F=FEPOINT, N=', I7, ', E=', I7,
F          ', ET=QUADRILATERAL, D=(1,2,3,4,5,6,7,8,9)')
        DO 4713 N=1,NUMELS
          IEL = IEL + 1
          l=nshpnt(n)
          if (ixs(4,l).eq.ixs(5,l))then
            write(31, '(4I8)') (ixs(i,l), i=2,4), 0
          else
            WRITE(31, '(4I8)') (ixs(i,l), i=2,5)
          endif
4713 CONTINUE
        ENDIF
C
        IEL = NUMELH + NUMELB + NUMELS
        IF (NUMELT .GE. 1) THEN
          DO 4714 N=1,NUMELT
            IEL = IEL + 1
            l=ntxpnt(n)
            WRITE(31, '(8I8)') (ixt(i,l), i=2,9)
4714 CONTINUE
          ENDIF
          CLOSE (UNIT=31)
C
        RETURN
      END

```


APPENDIX D. DOCUMENTATION PAGE FOR NEW MATERIAL TYPE

Material Type 44

Elastic-Plastic with Void Growth and Nucleation

Columns	Quantity	Format
Card 3		
1-10	Young's Modulus (E)	E10.0
11-20	Poisson's Ratio (ν)	E10.0
21-30	Initial Porosity/Void Content (Φ_0)	E10.0
31-40	Initial Yield Stress (S_{yp})	E10.0
Card 4		
Constants for Gurson's Void Model		
1-10	q_1 (Default = 1.5)	E10.0
11-20	q_2 (Default = 1.0)	E10.0
21-30	q_3 (Default = q_1^2)	E10.0
Constants for Void Nucleation Model		
31-40	Void Nucleation Particle Content (f_N) (Default = 0.0)	E10.0
41-50	Mean Nucleating Strain (e_N) (Default = 0.0)	E10.0
51-60	Nucleating Strain Standard Deviation (s_N) (Default = 0.0)	E10.0
61-70	Number of Strain-Hardening Line Segments Defined (iseg)	E10.0
Card 5		
Upper Strain Points of Line Segments		
1-10	End Point of 1st segment (ϵ_1)	E10.0
11-20	ϵ_2	E10.0
21-30	ϵ_3	E10.0
31-40	ϵ_4	E10.0
41-50	ϵ_5	E10.0
51-60	ϵ_6	E10.0
61-70	ϵ_7	E10.0
71-80	ϵ_8	E10.0
Card 6		
Upper Strain Points of Line Segments, Part 2		
1-10	ϵ_9	E10.0
11-20	ϵ_{10}	E10.0
21-30	ϵ_{11}	E10.0
31-40	ϵ_{12}	E10.0
41-50	ϵ_{13}	E10.0
51-60	ϵ_{14}	E10.0
61-70	ϵ_{15}	E10.0
71-80	ϵ_{16}	E10.0

Card 7

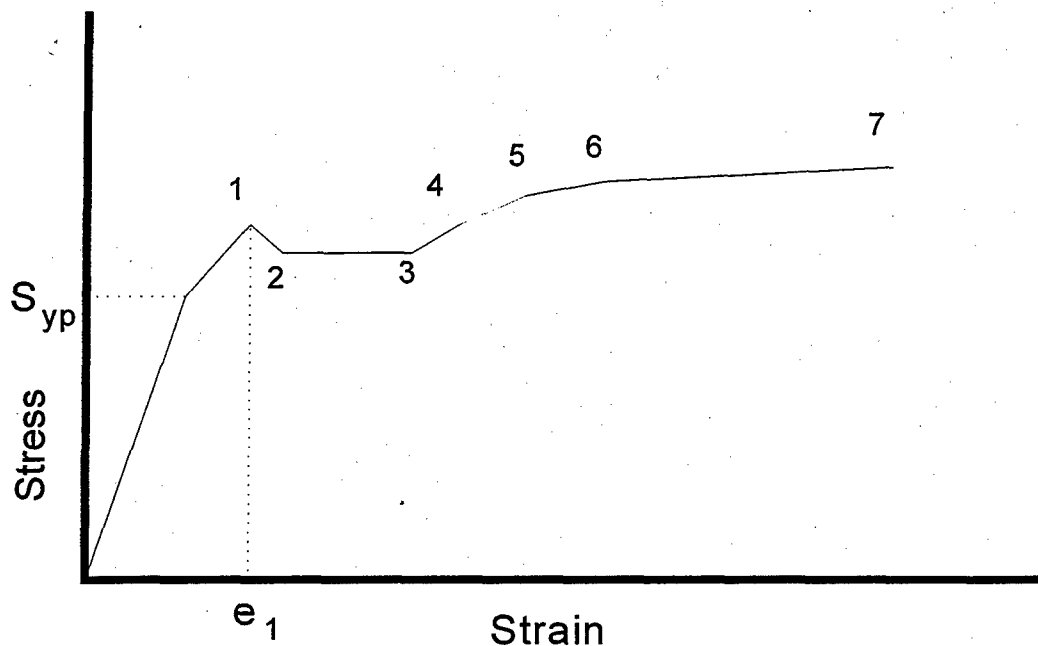
Stress End-points of Line Segments

1-10	Stress at End Point of 1st segment (σ_1)	E10.0
11-20	σ_2	E10.0
21-30	σ_3	E10.0
31-40	σ_4	E10.0
41-50	σ_5	E10.0
51-60	σ_6	E10.0
61-70	σ_7	E10.0
71-80	σ_8	E10.0

Card 8

Upper Strain Points of Line Segments, Part 2

1-10	σ_9	E10.0
11-20	σ_{10}	E10.0
21-30	σ_{11}	E10.0
31-40	σ_{12}	E10.0
41-50	σ_{13}	E10.0
51-60	σ_{14}	E10.0
61-70	σ_{15}	E10.0
71-80	σ_{16}	E10.0



As shown in the figure above, the stress and strain values are taken directly from standard tensile-test results for the material.

LIST OF REFERENCES

- [1] B. L. Koziey and F. A. Mirza, "Consistent Thick Shell Element," *Comput. Struct.* **65.4**, 531-549 (1997).
- [2] D. J. Allman, Evaluation of the Constant Strain Triangle with Drilling Rotations, *Int. J. Numer. Methods Engrg.* **26**, 2645-2655 (1988).
- [3] A. M. Khaski and A. L. Soler, "A Finite Element for General Thick Walled Shell Structures," *J. Press. Vessel Tech.* **107**, 126-133 (1985).
- [4] R. D. Cook, "Four-Node 'Flat' Shell Element: Drilling Degrees of Freedom, Membrane-Bending Coupling, Warped Geometry, and Behavior," *Comput. Struct.* **50.4**, 549-555 (1994).
- [5] B. L. Kemp, C. Cho, and S. W. Lee, "A Four-Node Solid Shell Element Formulation with Assumed Strain," *Int. J. Numer. Methods Engrg.* **43**, 909-924 (1998).
- [6] Y. Zhu and T. Zacharia, "A New One-Point Quadrature, Quadrilateral Shell Element with Drilling Degrees of Freedom," *Comp. Methods Appl. Mech. Engrg.* **136**, 165-203 (1996).
- [7] Q. Zeng and A. Comescure, "A New One-Point Quadrature, General Non-linear Quadrilateral Shell Element with Physical Stabilization," *Int. J. Numer. Methods Engrg.* **42**, 1307-1338 (1998).
- [8] G. Rengarajan, M. A. Aminpour, and N. F. Knight, Jr., "Improved Assumed-Stress Hybrid Shell Element with Drilling Degrees of Freedom for Linear Stress, Buckling and Free Vibration Analysis," *Int. J. Numer. Methods Engrg.* **38**, 1917-1943 (1995).
- [9] D. Briassoulis, "The C^0 Shell Plate and Beam Elements Freed from their Deficiencies," *Comp. Methods Appl. Mech. Engrg.* **72**, 243-266 (1989).
- [10] A. Ibrahimbegovic, R. L. Taylor, and E. L. Wilson, "A Robust Quadrilateral Membrane Finite Element with Drilling Degrees of Freedom," *Int. J. Numer. Methods Engrg.* **30**, 445-457 (1990).
- [11] S. W. Key and C. C. Hoff, "An Improved Constant Membrane and Bending Stress Shell Element for Explicit Transient Dynamics," *Comp. Methods Appl. Mech. Engrg.* **124**, 33-47 (1995).

- [12] K. S. Lay, "Shell Finite Element Formulated on Shell Middle Surface," *J. Engrg Mech.* **119.10**, 1973-1992 (1993).
- [13] F. L. Addessio, J. N. Johnson, and P. J. Maudlin, "The Effect of Void Growth on Taylor Cylinder Impact Experiments," *J. Appl. Physics* **73.11**, 7288-7297 (1993).
- [14] J. W. Hancock and R. D. Thomson, "Strain and Stress Concentrations in Ductile Fracture by Void Nucleation, Growth and Coalescence," *Mat. Sci. Tech.* **1**, 684-690 (1985).
- [15] C. C. Chu and A. Needleman, "Void Nucleation Effects in Biaxially Stretched Sheets," *J. Engrg. Mat. Tech.* **102**, 249-256 (1980).
- [16] V. Tvergaard, "Influence of Voids on Shear Band Instabilities under Plane Strain Conditions," *Int. J. Fracture* **17.4**, 389-406 (1981).
- [17] A. L. Gurson, "Continuum Thoery of Ductile Rupture by Void Nucleation and Growth: Part I – Yield Criteria and Flow Rules for Porous Ductile Materials," *J. Engrg. Mat. Tech.* **99**, 2-15 (1977).
- [18] J. C. Nagtegaal, D. M. Parks, and J. R. Rice, "On Numerically Accurate Finite Element Solutions in the Fully Plastic Range," *Comput. Methods Appl. Mech. Engrg.* **4**, 153-177 (1974).
- [19] G. Shi and G. Z. Voyiadjis, "A Computational Model for FE Ductile Plastic Damage Analysis and Plate Bending," *J. Appl. Mech.* **60**, 749-758 (1993).
- [20] J. H. Lee and Y. Zhang, "A Finite-Element Work-Hardening Plasticity Model of the Uniaxial Compression and Subsequent Failure of Porous Cylinders Including Effects of Void Nucleation and Growth – Part I: Plastic Flow and Damage," *J. Engrg. Mat. Tech.* **116**, 69-79 (1994).
- [21] N. Aravas, "On the Numerical Integration of a Class of Pressure-Dependent Plasticity Models," *Int. J. Numer. Methods Engrg.* **24**, 1395-1416 (1987).
- [22] T. J. R. Hughes and F. Brezzi, "On Drilling Degrees of Freedom," *Comput. Methods Appl. Mech. Engrg.* **72**, 105-121 (1989).
- [23] F. Essenburg, "On the Significance of the Inclusion of the Effect of Transverse Normal Strain in Problems Involving Beams With Surface Constraints," *J. Appl. Mech.* 127-132 (March 1975).
- [24] R. D. Cook, *Concepts and Applications of Finite Element Analysis*, 2nd ed., Wiley, New York, 1981.

- [25] T. Belytschko, C. S. Tsay, and W. K. Liu, "A Stabilization Matrix for the Bilinear Mindlin Plate Element," *Comput. Methods Appl. Mech. Engrg.* **29**, 313-327 (1981).
- [26] R. H. MacNeal and R. L. Harder, "A Proposed Set of Problems to Test Finite Element Accuracy," *Finite Elements in Design and Analysis*, ed. W. D. Pilkey, Vol. 1, North-Holland, Amsterdam, 3-20 (1985).
- [27] P.M. McDermott and Y.W. Kwon, "Development of a Shell Element with Pressure Variation through the Thickness - Part II," NPS-ME-98-005 Progress Report (1998).

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
8725 John J. Kingman Rd., STE 0944
Ft. Belvoir, Virginia 22060-6218

2. Dudley Knox Library, Code 013..... 2
Naval Postgraduate School
411 Dyer Rd.
Monterey, California 93943-5101

3. Professor Young W. Kwon, Code ME/Kw 4
Naval Postgraduate School
Monterey, California 93943

4. LT Patrick M. McDermott, USN 2
80 La Havre Circle
Florissant, MO 63031

5. Mr. John McKirgan, Code 614 1
Naval Surface Warfare Center
Carderock Division
9500 MacArthur Boulevard
West Bethesda, Maryland 20817-5700

6. Mr. Edward Johnson 1
Naval Surface Warfare Center
Indian Head Division
Code 40E, Bldg 302
Indian Head, Maryland 20640-5035

7. Research Office, Code 09 1
Naval Postgraduate School
Monterey, California 93942